(12) **United States Patent**
Gourlay et al.

(10) **Patent No.:** **US 9,483,879 B2**
(45) **Date of Patent:** **Nov. 1, 2016**

(54) **USING FREE-FORM DEFORMATIONS IN SURFACE RECONSTRUCTION**

(71) Applicant: **MICROSOFT TECHNOLOGY LICENSING, LLC**, Redmond, WA (US)

(72) Inventors: **Michael Jason Gourlay**, Seattle, WA (US); **Justin Avram Clark**, Kirkland, WA (US)

(73) Assignee: **MICROSOFT TECHNOLOGY LICENSING, LLC**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 139 days.

(21) Appl. No.: **14/489,802**

(22) Filed: **Sep. 18, 2014**

(65) **Prior Publication Data**

US 2016/0086385 A1 Mar. 24, 2016

(51) **Int. Cl.**
G06T 7/00	(2006.01)
G06T 19/20	(2011.01)
G06T 17/00	(2006.01)

(52) **U.S. Cl.**
CPC *G06T 19/20* (2013.01); *G06T 7/0065* (2013.01); *G06T 7/0077* (2013.01); *G06T 17/00* (2013.01); *G06T 2207/10004* (2013.01); *G06T 2207/20228* (2013.01); *G06T 2219/2021* (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2006/0290693 A1*	12/2006	Zhou	....	G06T 17/20 345/420
2010/0205126 A1*	8/2010	Andersen	....	G06F 17/10 706/13
2012/0306847 A1	12/2012	Lim et al.		
2015/0145985 A1*	5/2015	Gourlay	....	G06T 17/00 348/135
2015/0178988 A1*	6/2015	Montserrat Mora	...	G06T 13/40 345/420

OTHER PUBLICATIONS

Johannsson, et al, "Temporally scalable visual SLAM using a reduced pose graph", In 2013 IEEE International Conference on Robotics and Automation (ICRA), May 6, 2013, pp. 54-61.
"International Search Report and Written Opinion Issued in PCT Application No. PCT/US2015/049274", Mailed Date: Jan. 7, 2016, 9 Pages.
Roth, et al, "Moving Volume KinectFusion", Retrieved from <<http://www.bmva.org/bmvc/2012/BMVC/paper112/paper112. pdf>>, 2012, 11 Pages.
Whelan, et al, "Kintinous: Spatially Extended KinectFusion", Retrieved from <<http://dspace.mit.edu/handle/1721.1/71756>>, Jul. 19, 2012, 8 Pages.

(Continued)

Primary Examiner — Delomia L Gilliard
(74) *Attorney, Agent, or Firm* — Daniel Choi; Micky Minhas

(57) **ABSTRACT**

Volumes of a 3D physical space are used in a surface reconstruction process, where adjacent volumes share vertices so that no gaps or overlaps between the volumes exist. As a result, a continuous surface is obtained in the surface reconstruction process. The vertices are anchored to nodes in a pose graph, such that locations of the vertices are adjusted as the pose graph is updated. As a result, a deformation of the volumes is permitted. Based on the deformation of a volume, a region of a depth map of the physical space is deformed correspondingly. Each vertex can be anchored to a closest node of the pose graph, or to a point which is based on a combination of nodes. In one approach, the point is defined based on the closest node and other nodes within a defined radius of the closest node.

20 Claims, 22 Drawing Sheets

(56) **References Cited**

OTHER PUBLICATIONS

U.S. Appl. No. 13/931,507, Ambrus, et al., "Space Carving Based on Human Physical Data", filed Jun. 14, 2013.

Izadi, et al., "Kinect Fusion: Real-time 3D reconstruction and Interaction Using a Moving Depth Camera", In Proceedings of the 24th annual ACM symposium on User interface software and technology, Oct. 2011, 10 pages.

Zhu, et al., "Textured Mesh Surface Reconstruction of Large Buildings with Multi-View Stereo", In Journal of the Visual Computer, vol. 29, Issue 6-8, Jun. 2013, pp. 1-9.

Hilton, et al., "Multi-Resolution Geometric Fusion", In Proceedings of International Conference on Recent Advances in 3-D Digital Imaging and Modelling, May 12, 1997, 8 pages.

Wang, et al., "On the Structure of Nonlinearities in Pose Graph SLAM", In Proceedings of Robotics: Science and Systems VIII, Jul. 9, 2012, 8 pages.

Steinbrucker, et al., "Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences", In Proceedings of IEEE International Conference on Computer Vision, Dec. 1, 2013, pp. 4321-4328.

Zhou, et al., "Dense Scene Reconstruction with Points of Interest", In Journal of ACM Transactions on Graphics, vol. 32, Issue 4, Jul. 2013, 8 pages.

Marton, et al., "On Fast Surface Reconstruction Methods for Large and Noisy Point Clouds", In Proceedings of IEEE International Conference on Robotics and Automation, May 12, 2009, 6 pages.

Gain, James Edward, "Enhancing Spatial Deformation for Virtual Sculpting," A dissertation, St. John's College, University of Cambridge, Jun. 2000, 177 pages.

Sederberg, T.W., "Chapter 14: Free-Form Deformation (FFD)," Computer Aided Geometric Design Court Notes, Oct. 23, 2012, 6 pages.

U.S. Appl. No. 14/091,262, Gourlay et al., "Large-Scale Surface Reconstruction That is Robust Against Tracking and Mapping Errors", filed Nov. 26, 2013.

* cited by examiner

Fig. 1

```
                                        5

        ┌─────────────────────────┐
        │        Server, 10       │
        │  ┌───────────────────┐  │
        │  │   Communication   │  │
        │  │   interface, 11   │  │
        │  ├───────────────────┤  │
        │  │ Control circuits, 12 │
        │  ├───────────────────┤  │
        │  │    Memory, 13     │  │
        │  ├───────────────────┤  │
        │  │   Database, 14    │  │
        │  └───────────────────┘  │
        └─────────────────────────┘
                    ↕
               ╭─────────╮              ┌──────────────────┐
               │         │              │  Hub computing   │
               │   30    │  ←──────→    │   system, 50     │
               │         │              └──────────────────┘
               ╰─────────╯
                    ↕
        ┌─────────────────────────────────────┐
        │               Rig, 20               │
        │  ┌──────────────┐  ┌──────────────┐ │
        │  │Communication │  │Depth sensor, 25│
        │  │ interface, 21│  └──────────────┘ │
        │  └──────────────┘  ┌──────────────┐ │
        │  ┌──────────────┐  │Gaze detection│ │
        │  │  Augmented   │  │  system, 26  │ │
        │  │reality projection │ └──────────────┘ │
        │  │  system, 22  │  ┌──────────────┐ │
        │  └──────────────┘  │ Orientation  │ │
        │  ┌──────────────┐  │  sensor, 27  │ │
        │  │Control circuits, │ └──────────────┘ │
        │  │     23       │  ┌──────────────┐ │
        │  └──────────────┘  │  Memory, 28  │ │
        │  ┌──────────────┐  └──────────────┘ │
        │  │ Visible light│                    │
        │  │  camera, 24  │                    │
        │  └──────────────┘                    │
        └─────────────────────────────────────┘
```

Fig. 2

Fig. 3

Fig. 4

External microphone connector, 442

Band Interface, 402

Microphone Interface, 440

GPU, 422

RAM, 426

Display Out, 436

Display In, 438

CPU, 420

Cache, 424

Flash Controller, 432

Memory Controller, 428

PCI express, 444

USB Port, 448

404

406

HMD Power, 476

Clock Generator, 460

ADC Converter, 462

Battery Charger, 464

Voltage Regulator, 466

Flash Memory, 434

Memory, 430

Wireless Comm., 446

Charging Jack, 470

Battery, 468

105

500

CENTRAL PROCESSING UNIT 501

| LEVEL 1 CACHE 510 | LEVEL 2 CACHE 512 |

| FLASH ROM MEMORY 503 | MEMORY CONTROLLER 502 |

RAM MEMORY 506

3D GRAPHICS PROCESSING UNIT 520

VIDEO ENCODER 522

A/V PORT 528

AUDIO PROCESSING UNIT 524

AUDIO CODEC 526

USB HOST CONTROLLER 531

NW I/F 532

514

ATA CABLE 516

PORTABLE MEDIA DRIVE 505

HARD DISK DRIVE 508

APP 560

544

SYSTEM POWER SUPPLY MODULE 550

FAN 552

MICROCONTROLLER UNIT 554

OPTICAL I/O INTERFACE 535

FRONT PANEL I/O SUBASSEMBLY 542

WIRED/WIRELESS CONTROLLER PORT SUBASSEMBLY 540

MEMORY UNIT PORT "A" 530(1)

MEMORY UNIT PORT "B" 530(2)

544

504(2)

504(4)

540(2)

512

541

CONTROLLER 504(1)

CONTROLLER 504(3)

MEM. UNIT

540(4)

540(6)

REMOTE CONTROL 590

MEM. UNIT

MEM. UNIT

MEM. UNIT

540(3)

540(5)

540(1)

FIG. 5

Fig. 6A

Rig moves in different poses in a physical space, 600

Obtain images of physical space from camera on rig, 601

Obtain orientation readings from orientation sensor on rig, 602

Obtain depth maps of volumes of the physical space from depth sensor on rig, 604

Process images and orientation readings to provide and update a pose graph; maintain the shared vertices by allowing deformation of the volumes, 603

Partition the physical space into volumes, such that adjacent volumes have shared vertices; create a new volume if appropriate, 605

Anchor vertices of the volumes to points in the pose graph, 606

Reconstruct surfaces in the physical space based on a state of the pose graph, 607

Fig. 6B

Input orientation reading and image of pose; determine image features, 610

New pose?, 611

no

yes

Add a new node to the pose graph, 612

Determine a coordinate system associated with the orientation of the pose, 613

Associate the image taken with the pose and the coordinate system with the new node, 614

Determine a correspondence between the image associated with the new node and images of one or more other nodes, 615

Determine a transform between the coordinate system of the new node and coordinate systems of the one or more other nodes, 616

Associate the transform and the correspondence with one or more links between the new node and the one or more other nodes, 617

Update other nodes and links in the pose graph if appropriate, 618

Fig. 6C

Select a vertex, 620

Determine closest node, 621

Anchor vertex to the closest node, 622

Identify other nodes within radius of closest node, 625

Determine a point which is a combination of the closest node and the other nodes, 626

Anchor vertex to the point, 627

Next vertex in volume?, 623

yes

no

End, 624

Fig. 6D

Identify one or more volumes and associated depth maps for surface reconstruction, 630

Determine one or more points in the pose graph to which the vertices of the one or more volumes are anchored, 631

Determine the coordinate systems of the one or more points, 632

Project the one or more volumes into the one or more depth maps based on the one or more coordinate systems, 633

Fig. 7

Fig. 8A

Fig. 8B

Fig. 9A

Fig. 9B

Fig. 10

N1
N9
N2
L(N1,N3)
1210
1000
N3
L(N1,N2)
N8
N4
N7
N6
N5

Fig. 11A

```
Camera image: Image1
Orientation reading: OR1
```

Fig. 11B

```
Depth map: DM1
Orientation reading: OR1
```

```
Pose graph

Node id: N1
Camera Image:Image1
Coordinate system: CS1

Link id: L(N1,N2)
Image correspondence: C(N1,N2)
Transform between coordinate systems: T(N1,N2)

Link id: L(N1,N3)
Image correspondence: C(N1,N3)
Transform between coordinate systems: T(N1,N3)
```

Fig. 12A

```
Volume id: Vol1
Vertex id:      Anchored to:
v1              N1
v2              N2
v3              N3
v4              N4
v5              N5
v6              N6
v7              N7
v8              N8
```

Fig. 12B

```
Volume id: Vol2
Vertex id:      Anchored to:
v2              N2
v4              N4
v6              N6
v8              N8
v9              N9
v10             N10
v11             N11
v12             N12
```

Fig. 12C

```
Volume id: Vol1
Vertex id:      Anchored to:
v1              N1
v2              N2
v3              N3
v4              N4
v5              N5
v6              N5, N6, N11
v7              N7
v8              N8
```

Fig. 12D

```
Volume id: Vol2
Vertex id:      Anchored to:
v2              N2
v4              N4
v6              N5, N6, N11
v8              N8
v9              N9
v10             N10
v11             N11
v12             N12
```

Fig. 12E

Fig. 13A

depth map, 1315

wall, 1320

painting, 1321

doorway, 1322

Fig. 13B

Fig. 13C

Fig. 14A

camera image, 1400

Fig. 14B

Fig. 14C

camera image, 1440

Fig. 14D

Fig. 15

Fig. 16A

each vertex
linked to a node

Fig. 16B

one vertex linked to a point which is based on combination of nodes within a radius of a closest node
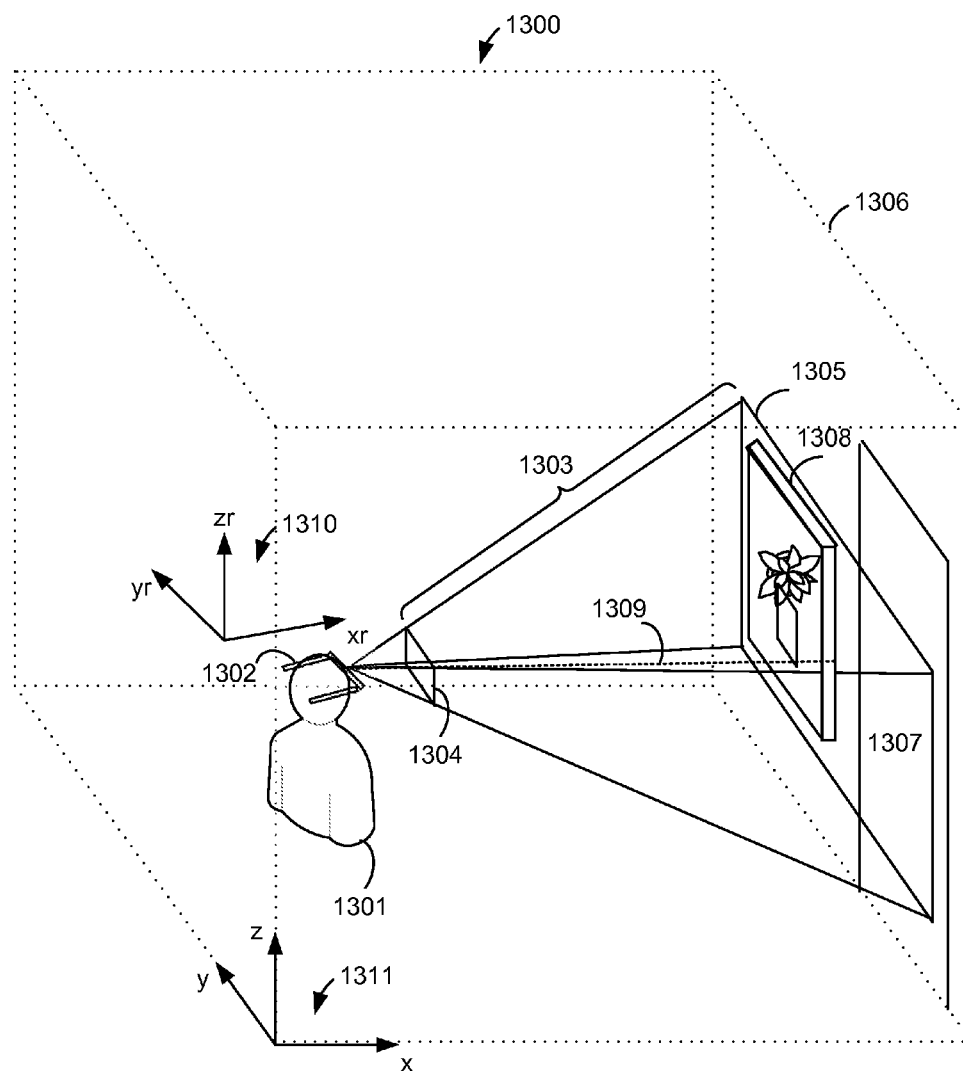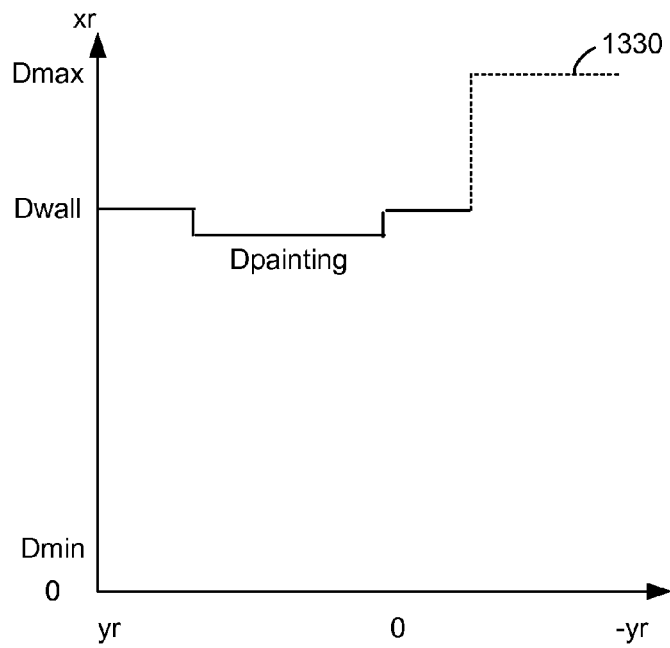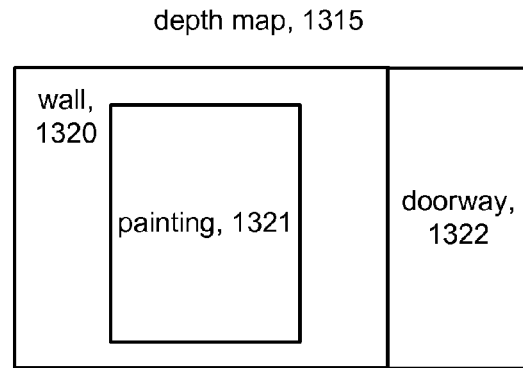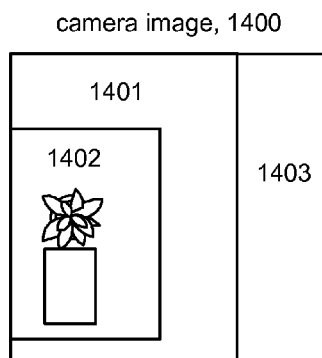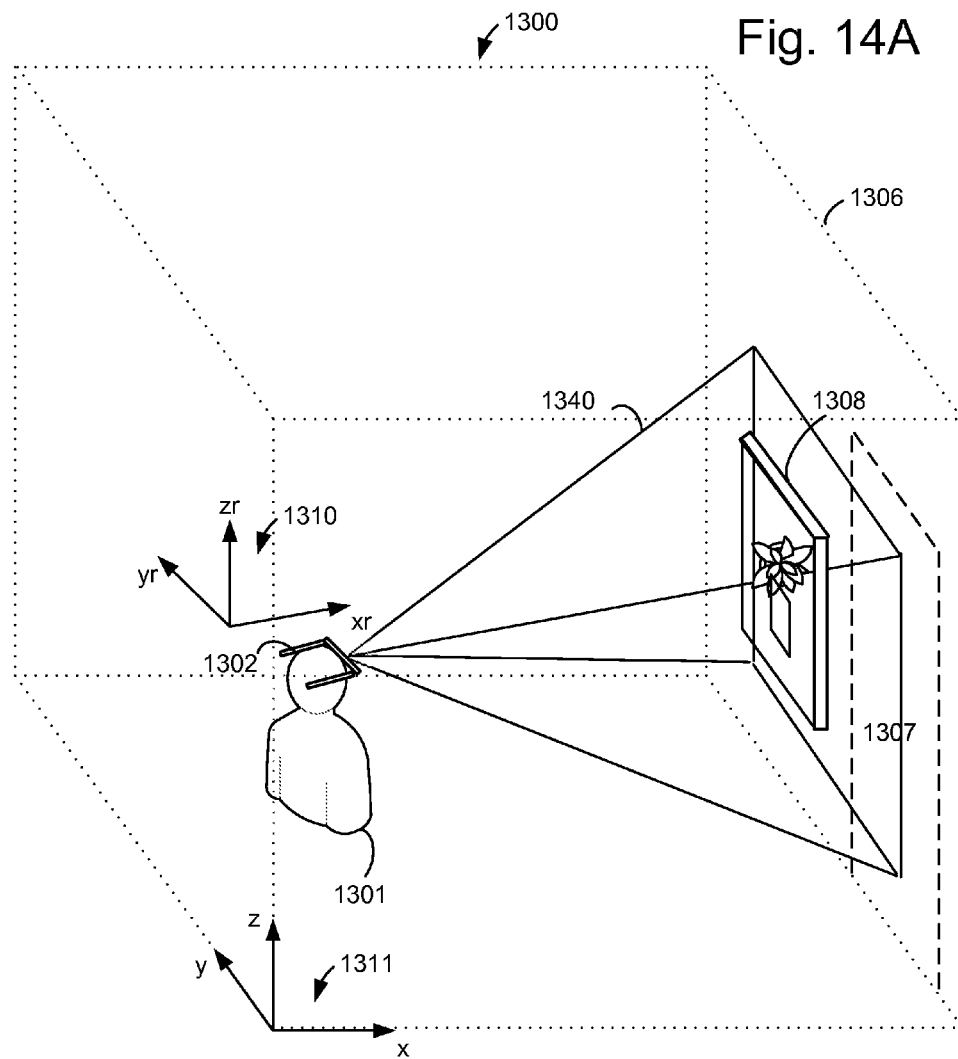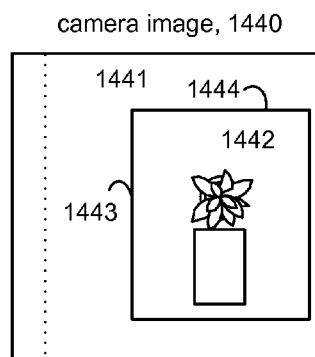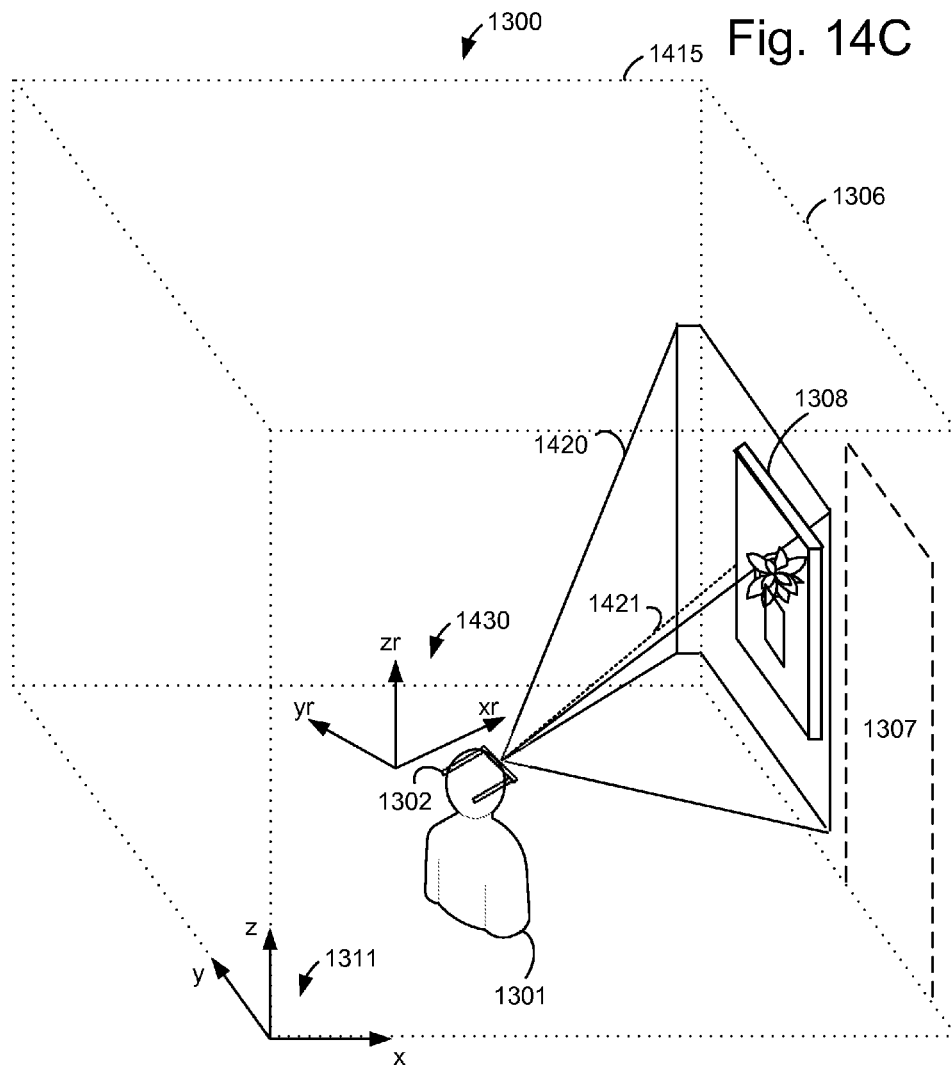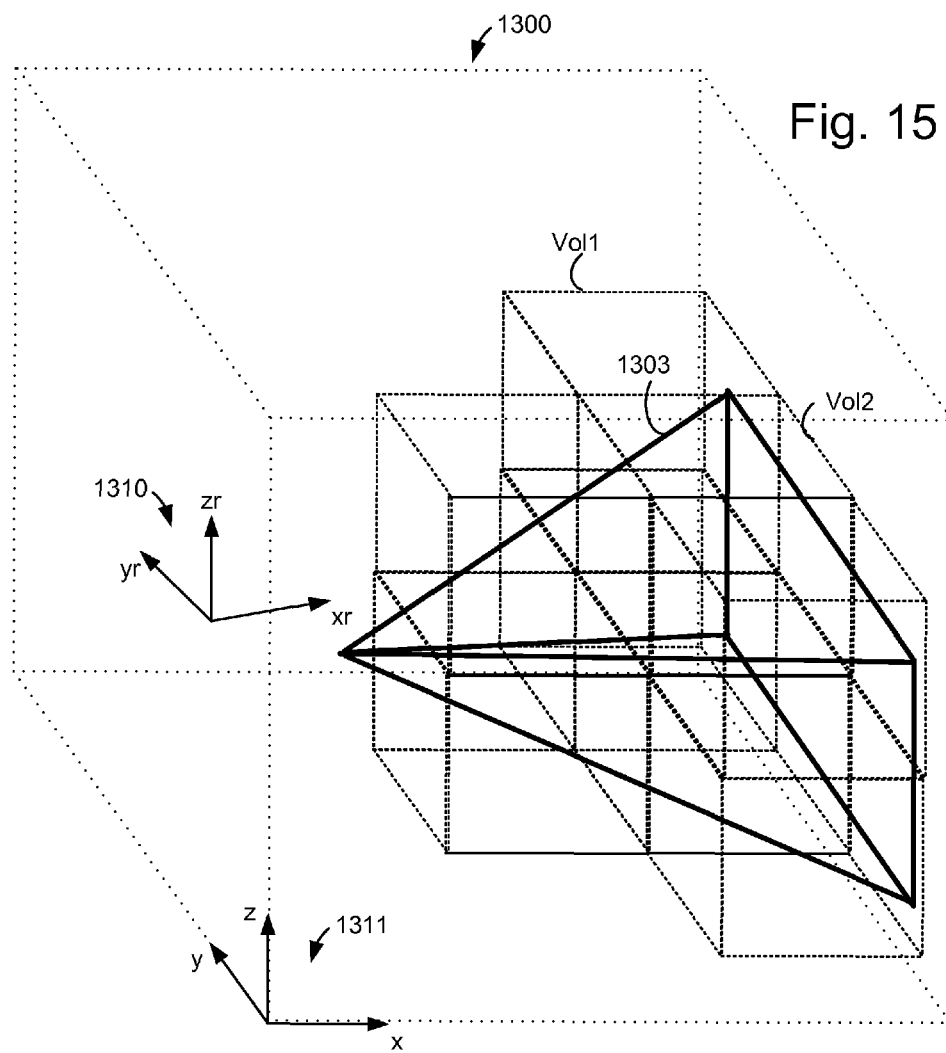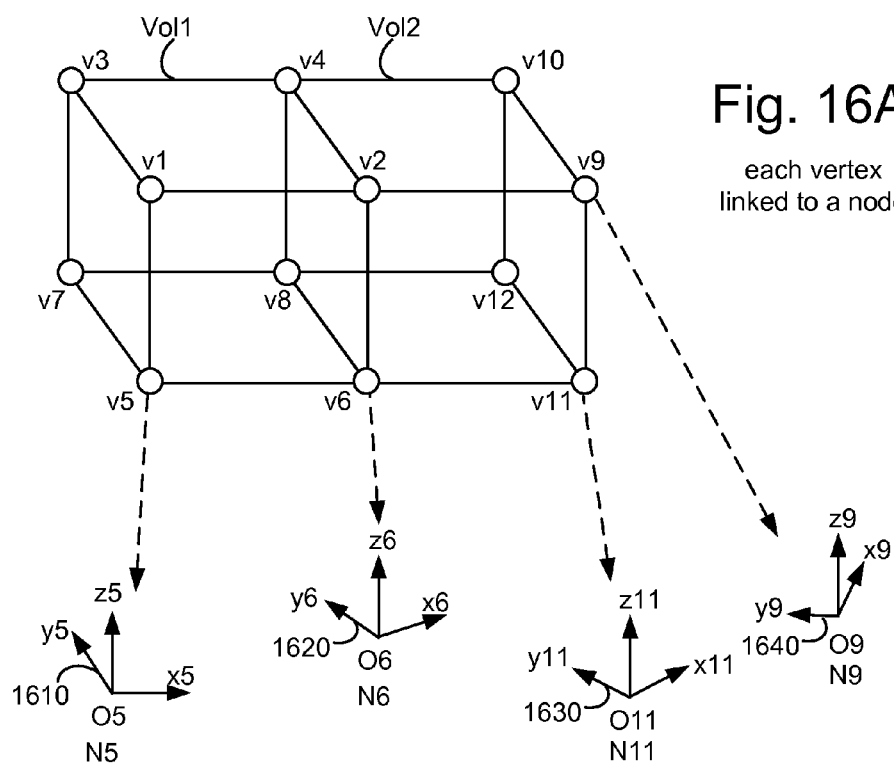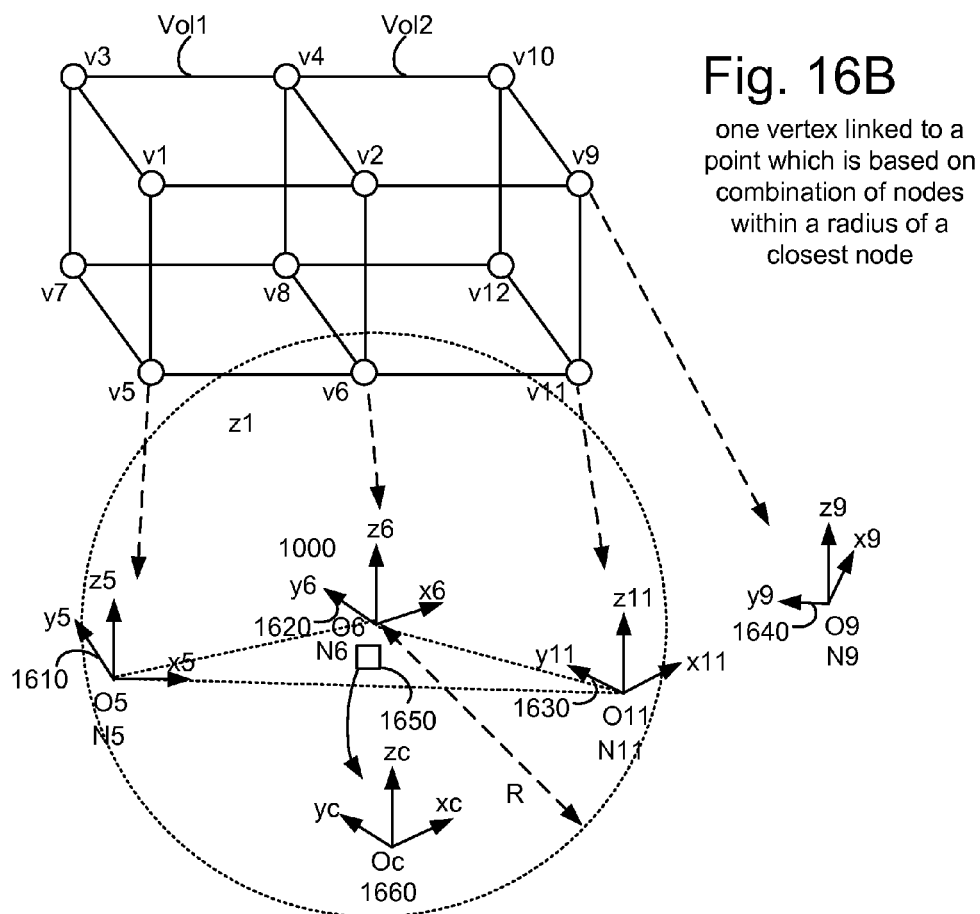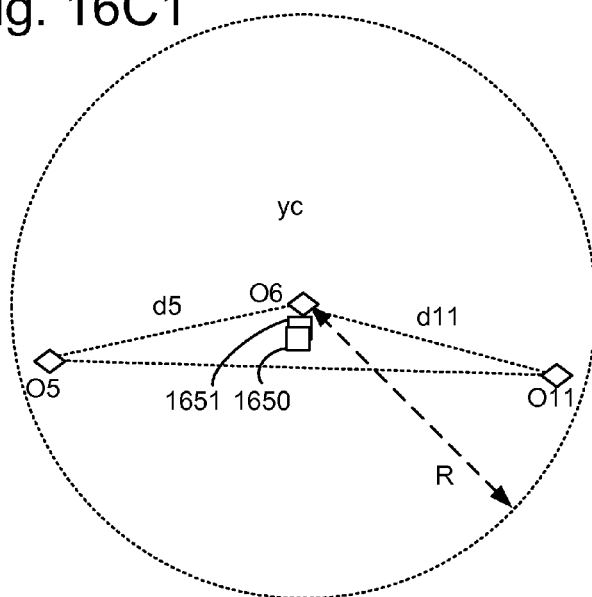
Fig. 16C1

Fig. 16C2

Fig. 16D

Fig. 16E

Fig. 17
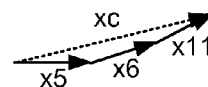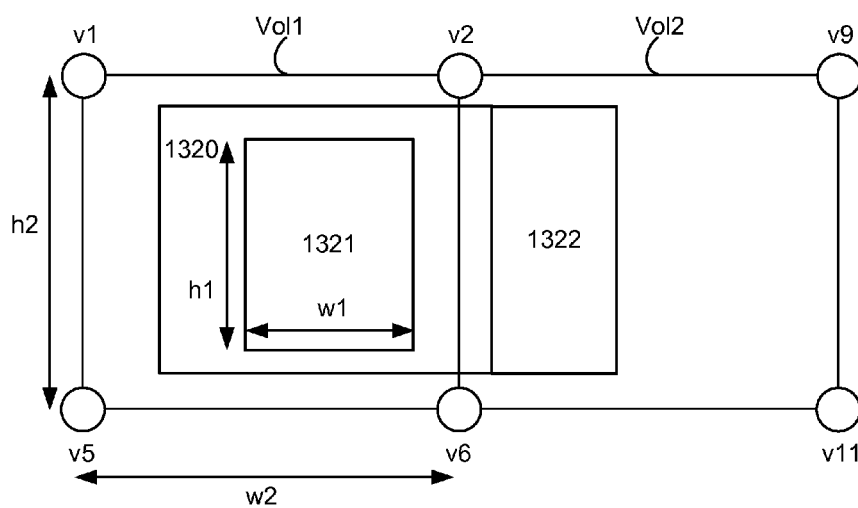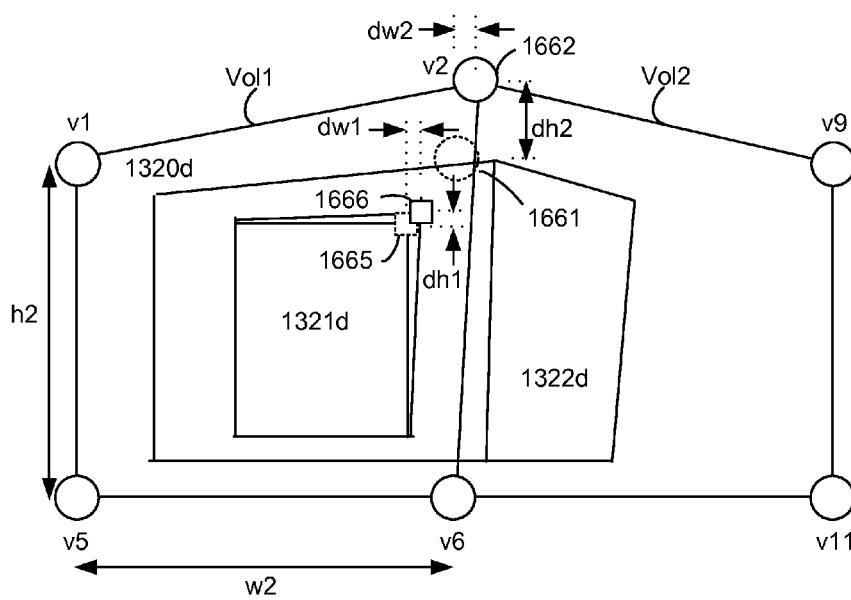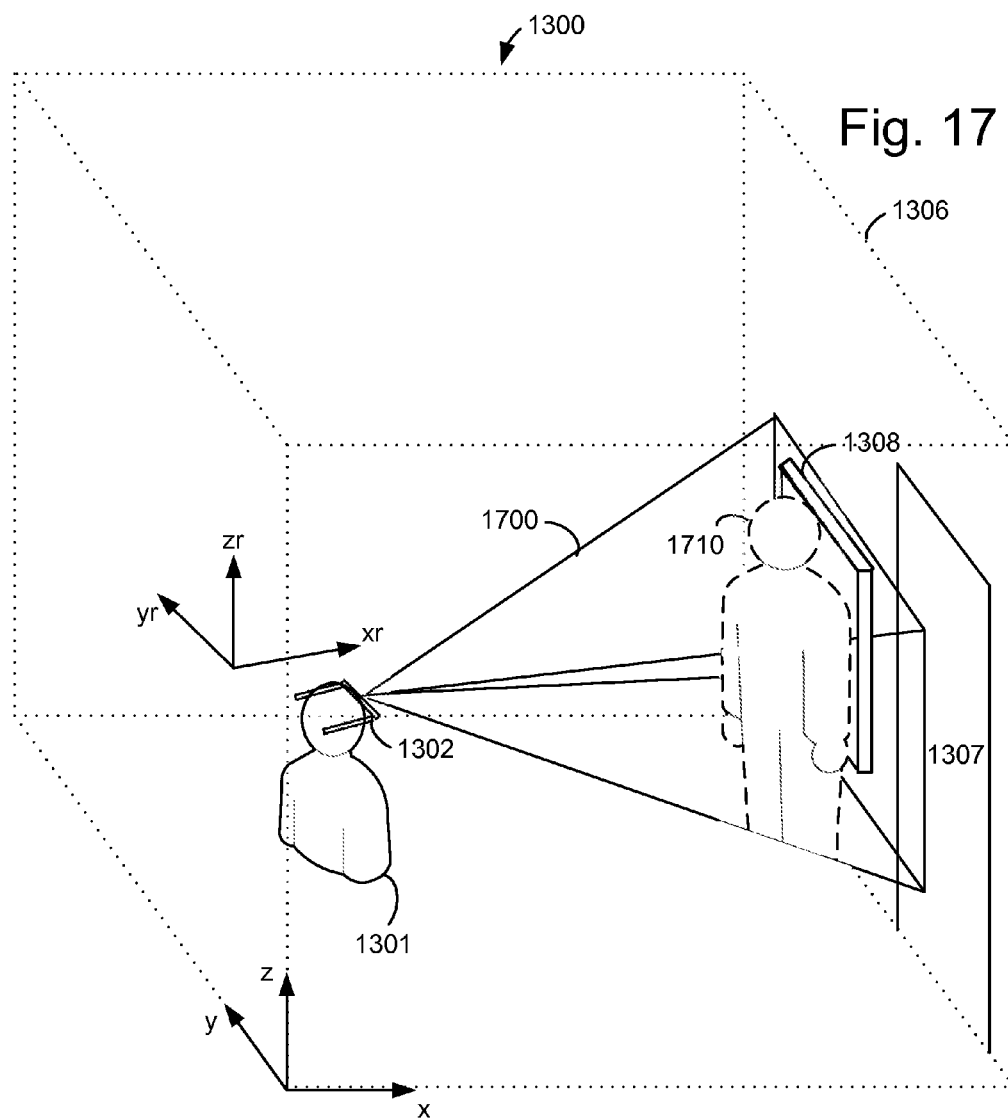
USING FREE-FORM DEFORMATIONS IN SURFACE RECONSTRUCTION

BACKGROUND

Surface reconstruction (SR) involves identifying surfaces of objects in a physical space. For example, the surfaces can represents walls, floors, a ceiling, furniture or other objects in a room. The surfaces provide a model of the physical environment. Such a model can be used in a wide range of applications. One example is guiding a robot through a physical space. Another example is displaying a virtual object to a user, such by using a head mounted display device.

SUMMARY

Technology described herein provides various embodiments for reconstructing surfaces in a physical space.

Depth maps of a physical space are obtained using a depth sensor. A depth map contains depth values which represent a distance from the depth sensor to surfaces in the physical space. The depth maps are defined with respect to volumes in the physical space, and the location of the volumes is defined with respect to a coordinate system. For example, the coordinate system can be defined according to a pose (e.g., position and orientation) of a rig on which the depth sensor is carried.

However, errors in the position of the volumes can accumulate as additional depth maps are obtained and additional volumes are defined. To reduce the accumulation of errors, visible light images of the physical space are obtained and used to form a pose graph. The pose graph comprises nodes which are associated with different poses of the rig and the corresponding images. Links between the nodes represent correspondences between the images, and transforms between coordinate systems of the nodes. From time to time, as new images are captured, the pose graph is updated. For example, a node can be updated by updating the coordinate system associated with the node.

Additionally, vertices, e.g., corners, of the volumes are anchored to or associated with different nodes, or combination of nodes, of the pose graph. In particular, adjacent volumes can have shared vertices on shared faces so that there is no gap or overlap between the volumes. When the pose graph is updated and the coordinate systems of the nodes change, the volumes are allowed to deform in shape so that the vertices of the shared faces continue to be anchored to the same nodes, or combination of nodes. Although, the vertices can be periodically re-anchored to a new node or combination of nodes if appropriate.

Since gaps and overlaps between adjacent volumes are avoided when a surface reconstruction process is performed, the surfaces in the different volumes will be smoothly joined across the volumes. Surfaces can be reconstructed at any time according to the current state of the pose graph.

This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, like-numbered elements correspond to one another.

FIG. **1** is a block diagram of a system **5** for surface reconstruction which includes a rig, a server and a hub computing system.

FIG. **2** is top view of an implementation of the rig **20** of FIG. **1** as a head mounted display (HMD) device **150**.

FIG. **3** is a block diagram depicting the various components of HMD device **150** of FIG. **2**.

FIG. **4** is a block diagram describing the various components of processing unit **105** of FIG. **2**.

FIG. **5** is a block diagram of one embodiment of the hub computing system **50** of FIG. **1**.

FIG. **6A** depicts a process for reconstructing surfaces in a physical space using a pose graph.

FIG. **6B** depicts further details of step **603** of FIG. **6A**.

FIG. **6C** depicts further details of step **606** of FIG. **6A**.

FIG. **6D** depicts further details of step **607** of FIG. **6A**.

FIG. **7** depicts a surface reconstruction pipeline for carrying out the process of FIG. **6A**.

FIG. **8A** depicts an example of a signed distance function (SDF) as discussed in connection with the 3D fusion subsystem **734** of FIG. **7**.

FIG. **8B** depicts an example of a depth map projected into voxels as discussed in connection with the 3D fusion subsystem **734** of FIG. **7**.

FIG. **9A** depicts a set of volumes **900-905** and a frustum **920** of a depth sensor.

FIG. **9B** depicts a new volume which is added to the set of volumes **900-905** of FIG. **9A**, as represented by an initially-placed volume **907** and a repositioned volume **906**.

FIG. **10** depicts an example of a pose graph which represents poses of a rig in a physical space.

FIG. **11A** depicts an example data record for an image captured by a camera.

FIG. **11B** depicts an example data record for a depth map captured by a depth sensor.

FIG. **12A** depicts an example data record of a pose graph.

FIG. **12B** depicts an example data record of a first volume in which each vertex of the volume is anchored to a different node.

FIG. **12C** depicts an example data record of a second volume which is adjacent to the first volume, and in which each vertex of the volume is anchored to a different node.

FIG. **12D** depicts an example data record of a first volume in which one vertex of the volume is anchored to a combination of nodes.

FIG. **12E** depicts an example data record of a second volume in which one vertex of the volume is anchored to a combination of nodes.

FIG. **13A** depicts an example physical space, showing a frustum of a depth sensor.

FIG. **13B** depicts a depth map consistent with FIG. **13A**.

FIG. **13C** depicts distance values of the depth map of FIG. **13B**.

FIG. **14A** depicts the example physical space of FIG. **13A**, showing a field of view of a visible light camera.

FIG. **14B** depicts an image obtained by the visible light camera in the pose of FIG. **14A**.

FIG. **14C** depicts the example physical space of FIG. **13A**, showing another field of view of the visible light camera as a pose of the HMD device is changed.

FIG. **14D** depicts an image obtained by the visible light camera in the pose of FIG. **14C**.

FIG. **15** depicts an initial placement of volumes in the physical space of FIG. **13**A.

FIG. **16**A depicts an example of step **606** of FIG. **6**A involving the volumes Vol1 and Vol2 of FIG. **15**, in which each vertex of a volume is linked to a different node, consistent with the data records of FIGS. **12**B and **12**C.

FIG. **16**B depicts an example of step **606** of FIG. **6**A involving the volumes Vol1 and Vol2 of FIG. **15**, in which a vertex of a volume is linked to combination of nodes, consistent with the data records of FIGS. **12**D and **12**E.

FIG. **16**C1 depicts distance d**5** and d**11** of the nodes N**5** and N**11**, respectively, from the closest node N**6** to v**6**.

FIG. **16**C2 depicts an x-axis of the combined coordinate system **1660** of FIG. **16**B.

FIG. **16**D depicts depth maps of undistorted objects within the volumes Vol1 and Vol2 when these volumes are not distorted. FIG. **16**E depicts depth maps of distorted objects within the volumes Vol1 and Vol2 when these volumes are distorted.

FIG. **17** depicts an example application in which a virtual object is placed in the physical space of FIG. **13**A.

DETAILED DESCRIPTION

Surface reconstruction can be used in many applications, including placing a virtual object, defining a play space, path-finding, collision detection or occlusion of virtual objects in a physical space.

Surface reconstruction over large scales requires robustness against tracking and mapping errors. For example, a physical space having a large scale can be a room in a home, office or museum, or an outdoor space. In a surface reconstruction process, surfaces are acquired using multiple depth maps which are acquired by a depth sensor carried on a rig. The surfaces are localized with respect to each other using computer vision tracking and mapping techniques which are prone to cumulative inaccuracies. Furthermore, pose estimates of the rig change over time as more information about the environment becomes available.

Some SR solutions operate on a small scale or ignore tracking errors. However, this results in SR data that is not self-consistent. Moreover, some SR solutions allow gaps or overlap between the volumes. Gaps result in a lack of surface data which would be clearly noticeable in a reconstructed surface. Overlaps result in redundant or conflicting data. Redundant data costs resources such as memory, computational time and power. Conflicting data can cause visible artifacts.

The techniques provided herein address the above and other issues. In one approach, an SR process involves creating multiple separate SR volumes to reconstruct. Each vertex of a volume can be anchored to (e.g., associated with) a node (e.g., a key frame), or a combination of nodes, in a tracking and mapping pose graph. Also, adjacent volumes can share vertices so that they have shared faces. In this case, the location of each vertex can be defined based on the coordinate system and pose of the respective node to which it is anchored. The locations of the vertices of a single volume can therefore be defined based on the coordinate systems and poses of different nodes. When the pose graph updates, such as due to a bundle adjustment or a re-localization, the vertexes remain anchored to their respective nodes. As a result, the volumes are allowed to deform and change their shape, and gaps and overlaps between volumes continue to be avoided. In practice, these changes in shape are relatively small so that they are not usually visible when SR is performed.

New volumes can be created to ensure coverage of the physical space as additional depth maps are obtained. Vertexes of the volumes can be occasionally re-anchored if appropriate.

FIGS. **1-5** describe various systems which can be used for surface reconstruction. FIG. **6** and later figures describe surface reconstruction techniques and scenarios.

FIG. **1** is a block diagram of a system **5** for surface reconstruction which includes a rig, a server and a hub computing system. The rig, server and hub computing system are examples of hardware computing devices. A rig **20** can represent a device which moves around in a physical space. The device may be mounted to an autonomous mobile robot, for instance, or carried or worn by a user, e.g., a person, such as on a head mounted display device, for instance. The rig has different poses as it moves around. In the context of tracking and mapping algorithms, a pose indicates a point in space and a direction. That is, it is a combination of a position and an orientation. The rig includes a communication interface **21**, such as for communicating with a server **10** via a network **30**. An augmented reality projection system **22** can be used for displaying augmented reality images (e.g., virtual objects) in the physical space in which surface reconstruction is performed. Control circuits **23** are used for controlling the various components of the rig. A visible light camera **24** obtains images of the physical space, e.g., at a fixed rate such as several times per second, or at specified times.

A depth sensor **25** obtains a depth map of the physical space. Typically, the depth maps are obtained less frequently than the camera images. The depth sensor can be a time-of-flight camera or a stereoscopic camera, for instance. A gaze detection system **26** may be used to determine the user's gaze in the case of the rig being a head mounted display device. For example, this can be useful in placing virtual objects in the physical space. An orientation sensor **27** obtains readings regarding an orientation of the rig. For, example, an orientation reading can be associated with each image and depth map. These can be inertial sensors, for instance, discussed further below. A memory **28** can store executable code which is executed by the control circuits to provide the functionality described herein, as well as storing information which is obtained by the visible light camera **24**, the depth sensor **25**, the gaze detection system **26** and the orientation sensor **27**.

Optionally, the rig communicates with a server **10**. For example, the rig could upload some of the information that it gathers or download information such as executable code, data for the augmented reality projection system, or other content. In one approach, the server is remote from the rig. In another approach, the rig communicates with a local hub computing system **50**, such as in the user's home. The hub computing system could be a gaming console which runs various gaming and non-gaming applications, for instance. See also FIG. **5**.

In one approach, the network **30** is a wireless network such as WI-FI®, BLUETOOTH®, infra-red (e.g., IrDA®) or cellular (e.g., GSM). Optionally, multiple rigs can communicate with one another within a common physical space.

Generally, the communication interfaces allow communication between computing devices. The control circuits provide control of hardware and/or software of the respective computing devices. For example, the control circuits can include one or more processors which execute instructions, and are configured to execute instructions, stored on one or more tangible, non-transitory processor-readable storage devices, e.g., memories, having processor-readable

5

software embodied thereon for programming a processor to perform processor- or computer-implemented methods as described herein. The memories can store the instructions as code, and can provide the processor-readable storage devices. The memories can provide databases, data stores or other sources of data which are accessed to perform the techniques described herein. The memories can be hardware memory devices.

Alternatively, or in addition, the functionally described herein can be performed, at least in part, by one or more hardware logic components. For example, and without limitation, illustrative types of hardware logic components that can be used include Field-programmable Gate Arrays (FP-GAs), Application-specific Integrated Circuits (ASICs), Application-specific Standard Products (ASSPs), System-on-a-chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), etc.

Further details of these system are provided in connection with the following figures.

FIG. 2 is top view of an implementation of the rig 20 of FIG. 1 as a head mounted display (HMD) device 150. Only the right side of the HMD device is depicted in detail. The HMD device includes a head-mounted frame 101 which can be generally in the shape of an eyeglass frame, temples 102 and 103, and a front lens frame including a nose bridge 104. Built into nose bridge 104 is a microphone 110 for recording sounds, including spoken user commands, and transmitting corresponding audio data to processing unit 105. Lenses 116 and 119 are see-through, in an example of a see-through near-to-the-eye HMD device. An immersive HMD device could alternatively be used.

The HMD device can be worn on the head of a user so that the user can see through a display and thereby see a real-world scene which includes an image which is not generated by the HMD device. The HMD device can be self-contained so that all of its components are carried by the frame. Optionally, one or more components of the HMD device are not carried by the frame. For example, one of more components which are not carried by the frame can be physically attached by a wire to a component carried by the frame.

Further, one of more components which are not carried by the frame can be in wireless communication with a component carried by the frame, and not physically attached by a wire or otherwise to a component carried by the frame. The one or more components which are not carried by the frame can be carried by the user, in one approach, such as on the wrist. For example, the processing unit 105 could be connected to a component in the frame via a wire or via a wireless link. The term "HMD device" can encompass both on-frame components and associated off-frame components.

The processing unit 105 includes much of the computing power used to operate the HMD device. The processor may execute instructions stored on a processor readable storage device for performing the processes described herein. In one embodiment, the processing unit communicates wirelessly with one or more servers, hub computing systems, other HMDs or other computing devices. Control circuits 136 provide various electronics that support the other components of the HMD device.

At the front of HMD device are one or more forward- or room-facing visible light cameras 113. The cameras can include at least one visible light video camera that can capture video and still images, and transmit the images to the processing unit. The camera can be used to identify features in the physical space, such as colors, patterns, shapes, lines and so forth. A depth sensor can be formed by the combi-

6

nation of an infrared emitter 115 and an infrared sensor/detector 117. The visible light camera may also be part of the depth sensor. The visible light camera 113 faces outward and has a viewpoint similar to that of the user.

A portion of the frame of the HMD device surrounds a display that includes one or more lenses. The display includes a light guide optical element 112, opacity filter 114, a front, right-side see-through lens 116 and a rear, right-side see-through lens 118. In one embodiment, opacity filter 114 is behind and aligned with see-through lens 116, light guide optical element 112 is behind and aligned with opacity filter 114, and see-through lens 118 is behind and aligned with light guide optical element 112. Opacity filter 114 filters out natural light (either on a per pixel basis or uniformly) to enhance the contrast of the augmented reality imagery. Light guide optical element 112 channels artificial light to the eye. Similarly, the left side of the HMD includes a front, left-side see-through lens 119 and a rear, left-side see-through lens 121.

Mounted to or inside temple 102 is an image source, which (in one embodiment) includes microdisplay 120 for projecting an augmented reality image and lens 122 for directing images from the microdisplay into light guide optical element 112. In one embodiment, lens 122 is a collimating lens. An augmented reality emitter can include the microdisplay, one or more optical components such as the lens 122 and light guide optical element 112, and associated electronics such as a driver. Such an augmented reality emitter is associated with the HMD device, and emits light to a user's eye, where the light represents augmented reality still or video images. This can be used to display a virtual object in a physical space.

Control circuits 136, discussed further in connection with FIG. 3, provide various electronics that support the other components of HMD device. Inside, or mounted to temple 102, are ear phones 130 and inertial sensors 132 (e.g., an example of the orientation sensor 27 of FIG. 1). In one embodiment, the inertial sensors 132 include a three axis magnetometer 132A, three axis gyroscope 132B and three axis accelerometer 132C (see FIG. 3). The inertial sensors are for sensing position, orientation, sudden accelerations of HMD device. For example, the inertial sensors can be one or more sensors which are used to determine an orientation and location of the HMD worn on the user's head.

The microdisplay projects an image through lens 122. Light guide optical element 112 transmits light from the microdisplay to the eye 140 of the user while allowing light from in front of the HMD device to be transmitted through light guide optical element to eye 140, as depicted by arrow 142, allowing the user to have an actual direct view of the space in front of HMD device, in addition to receiving an augmented reality image from the microdisplay. Thus, the walls of light guide optical element are see-through. Light guide optical element includes a first reflecting surface 124 (e.g., a mirror or other surface). Light from the microdisplay passes through lens 122 and is incident on reflecting surface 124, which reflects the incident light such that light is trapped inside a planar, substrate comprising light guide optical element by internal reflection. After several reflections off the surfaces of the substrate, the trapped light waves reach an array of selectively reflecting surfaces, including example surface 126.

Reflecting surfaces 126 couple the incident light waves out of the substrate into the eye 140 of the user. Different light rays will travel and bounce off the inside of the substrate at different angles as they hit the various reflecting surface 126 at different angles. Therefore, different light rays

will be reflected out of the substrate by different ones of the reflecting surfaces. The selection of which light rays will be reflected out of the substrate by which surface 126 is engineered by selecting an appropriate angle of the surfaces 126. In one embodiment, each eye will have its own light guide optical element. When the HMD device has two light guide optical elements, each eye can have its own micro-display that can display the same image in both eyes or different images in the two eyes. In another embodiment, there can be one light guide optical element which reflects light into both eyes.

FIG. 3 is a block diagram depicting the various components of HMD device 150 of FIG. 2. The HMD device components include many sensors that track various conditions. In an example implementation, the HMD device can receive instructions about an augmented reality image from the processing unit 105 and provide the sensor information back to the processing unit, depicted in FIG. 4. Optionally, the processing unit also receives sensory information from a server or a hub computing device. Based on that information, and/or other criterion, the processing unit can determine where and when to provide an augmented reality image to the user and send instructions accordingly to the HMD device.

Note that some of the components (e.g., eye tracking camera 134B, microdisplay 120, opacity filter 114, eye tracking illumination 134A and earphones 130) are shown in shadow to indicate that there are two of each of those devices, one for the left side and one for the right side of HMD device. Similarly, the depth sensor 313 can include an infrared emitter 115 and an infrared sensor/detector 117, for instance. In another approach, two or more cameras with a known spacing between them are used as a depth camera to obtain depth data for objects in a room, indicating the distance from the cameras/HMD device to the object. Optionally, one visible light camera 113 is used. The eye tracking camera 134B and the eye tracking illumination 134A are part of an eye tracking component 134.

Images from the forward-facing cameras can be used to identify people, hand gestures and other objects in a field of view of the user. For example, it can be determined when the user makes a hand gesture such as pointing at an object. The real-world object can be identified and associated with a data stream, or used as the location to display a previously-associated data stream.

A control circuit 300 communicates with the power management circuit 302. Control circuit 300 includes processor 310, memory controller 312 in communication with memory 344 (e.g., DRAM), camera interface 316, camera buffer 318, display driver 320, display formatter 322, timing generator 326, display out interface 328, and display in interface 330. A GPS circuit 317 can be used to identify the location of the HMD device.

In one embodiment, all of components of control circuit 300 are in communication with each other via dedicated lines or one or more buses. In another embodiment, each of the components of control circuit 300 communicates with processor 310. A camera interface/buffer 316 provides an interface to, and stores images from, the visible light camera. A depth sensor interface/buffer 318 provides an interface to, and stores images from, the depth sensor.

Display driver 320 drives the microdisplay. Display formatter 322 provides information, about the augmented reality image being displayed on the microdisplay, to opacity control circuit 324, which controls opacity filter 114. Timing generator 326 is used to provide timing data for the system. Display out interface 328 is a buffer for providing images

from forward-facing cameras 113 to the processing unit 105. Display in interface 330 is a buffer for receiving images such as an augmented reality image to be displayed on the microdisplay.

Display out interface 328 and display in interface 330 communicate with band interface 332 which is an interface to processing unit, when the processing unit is attached to the frame of the HMD device by a wire, or communicates by a wireless link, and is worn on the wrist of the user on a wrist band. This approach reduces the weight of the frame-carried components of the HMD device. In other approaches, as mentioned, the processing unit can be carried by the frame and a band interface is not used.

Power management circuit 302 includes voltage regulator 334, eye tracking illumination driver 337, audio DAC and amplifier 338, microphone preamplifier audio ADC 340 and clock generator 345. Voltage regulator 334 receives power from processing unit via band interface 332 and provides that power to the other components of HMD device. Eye tracking illumination driver provides the infrared (IR) light source for eye tracking illumination 134A, as described above. Audio DAC and amplifier 338 receives the audio information from earphones 130. Microphone preamplifier and audio ADC 340 provides an interface for microphone 110. Power management unit 302 also provides power and receives data back from three-axis magnetometer 132A, three-axis gyroscope 132B and three axis accelerometer 132C, as part of an orientation sensor.

FIG. 4 is a block diagram describing the various components of the processing unit 105 of FIG. 2. Control circuit 404 communicates with power management circuit 406. Control circuit 404 includes a central processing unit (CPU) 420, graphics processing unit (GPU) 422, cache 424, RAM 426, memory control 428 in communication with memory 430 (e.g., DRAM), flash memory controller 432 in communication with flash memory 434 (or other type of non-volatile storage), display out buffer 436 in communication with HMD device via band interface 402 and band interface 332 (when used), display in buffer 438 in communication with HMD device via band interface 402 and band interface 332 (when used), microphone interface 440 in communication with an external microphone connector 442 for connecting to a microphone, Peripheral Component Interconnect (PCI) express interface 444 for connecting to a wireless communication component 446, and USB port(s) 448.

In one embodiment, wireless communication component 446 can include a Wi-Fi® enabled communication device, BLUETOOTH® communication device, or infrared communication device. The wireless communication component 446 is a wireless communication interface which, in one implementation, receives data in synchronism with the content displayed by the HMD device. Further, augmented reality images may be displayed in response to the received data. In one approach, such data is received from a server, a hub computing system, a local Wi-Fi® network or WLAN, a cell phone network, a local storage device or other source.

The USB port can be used to dock the processing unit to hub computing system 50 to load data or software onto processing unit, as well as charge processing unit. In one embodiment, CPU 420 and GPU 422 are the main work-horses for determining where, when and how to insert augmented reality images into the view of the user.

Power management circuit 406 includes clock generator 460, analog to digital converter 462, battery charger 464, voltage regulator 466 and HMD power source 476. Analog to digital converter 462 is connected to a charging jack 470 for receiving an AC supply and creating a DC supply for the

system. Voltage regulator 466 communicates with battery 468 for supplying power to the system. Battery charger 464 is used to charge battery 468 (via voltage regulator 466) upon receiving power from charging jack 470. HMD power source 476 provides power to the HMD device.

The calculations that determine where, how and when to insert an augmented reality image can be performed by the HMD device and/or by a computing device which communicates with the HMD device.

In one example embodiment, the HMD device will create a model of the environment that the user is in and track various objects in that environment, based on the field of view of the HMD device. The model and the tracking information are provided to processing unit. Sensor information obtained by HMD device is transmitted to processing unit. Processing unit refines the field of view of the user and provide instructions to HMD device on how, where and when to insert augmented reality images.

FIG. 5 is a block diagram of one embodiment of the hub computing system 50 of FIG. 1. In this embodiment, the computing system is a multimedia console 500 such as a gaming console. The multimedia console has a CPU 501, and a memory controller 502 that facilitates processor access to various types of memory, including a flash Read Only Memory (ROM) 503, a Random Access Memory (RAM) 506, a hard disk drive 508, and portable media drive 505. In one implementation, CPU includes a level 1 cache 510 and a level 2 cache 512, to temporarily store data and hence reduce the number of memory access cycles made to the hard drive 508, thereby improving processing speed and throughput.

CPU, memory controller, and various memory devices are interconnected via one or more buses (not shown).

In one implementation, CPU, memory controller, ROM, and RAM are integrated onto a common module 514. In this implementation, ROM is configured as a flash ROM that is connected to memory controller via a PCI bus and a ROM bus (neither of which are shown). RAM is configured as multiple Double Data Rate Synchronous Dynamic RAM (DDR SDRAM) modules that are independently controlled by memory controller via separate buses (not shown). Hard disk drive and portable media drive are shown connected to the memory controller via the PCI bus and an AT Attachment (ATA) bus 516.

A GPU 520 and a video encoder 522 form a video processing pipeline for high speed and high resolution graphics processing.

An audio processing unit 524 and an audio codec (coder/decoder) 526 form a corresponding audio processing pipeline for multi-channel audio processing of various digital audio formats. Audio data are carried between audio processing unit and the audio codec via a communication link (not shown). The video and audio processing pipelines output data to an A/V (audio/video) port 528 for transmission to a television or other display. In the illustrated implementation, video and audio processing components 520, 522, 524, 526 and 528 are mounted on the module.

The module 514 includes a USB host controller 531 and a network interface 532. USB host controller is shown in communication with CPU and memory controller via a bus (e.g., PCI bus) and serves as host for peripheral controllers 504(1)-504(4). Network interface provides access to a network (e.g., Internet, home network, etc.) and may be any of a wide variety of various wired or wireless interface components.

In the implementation depicted, the console includes a controller support subassembly 540 for supporting the four

peripheral controllers. The controller support subassembly includes any hardware and software components needed to support wired and wireless operation with an external control device, such as for example, a media and game controller. A front panel I/O subassembly 542 supports the multiple functionalities of power button 512, the eject button 541, as well as any LEDs (light emitting diodes) or other indicators exposed on the outer surface of console. Subassemblies and are in communication with module via one or more cable assemblies 544. In other implementations, the console can include additional controller subassemblies. An optical I/O interface 535 sends and receives signals that can be communicated to module 514. The interface may be responsive to a remote control 590.

Memory units (MUs) 540(1) and 540(2) are connectable to MU ports "A" 530(1) and "B" 530(2) respectively. Additional MUs (e.g., MUs 540(3)-540(6)) are illustrated as being connectable to the peripheral controllers 504(1) and 504(3), i.e., two MUs for each controller. Controllers 504(2) and 504(4) can also be configured to receive MUs (not shown). Each MU offers additional storage on which games, game parameters, and other data may be stored. In some implementations, the other data can include any of a digital game component, an executable gaming application, an instruction set for expanding a gaming application, and a media file. When inserted into the console or a controller, MU can be accessed by memory controller. A system power supply module 550 provides power to the components of the console. A fan 552 cools the circuitry within the console. A microcontroller unit 554 is also provided.

An application 560 comprising machine instructions is stored on hard disk drive. When the console is powered on, various portions of application are loaded into the RAM, and/or the caches, for execution on the CPU, wherein application is one such example. Various applications can be stored on hard disk drive for execution on CPU.

The console may be operated as a standalone system by simply connecting the system to a monitor, a television, a video projector, or other display device. In this standalone mode, the console enables one or more players to play games, or enjoy digital media, e.g., by watching movies, or listening to music. However, with the integration of broadband connectivity made available through network interface, the console may further be operated as a participant in a larger network gaming community.

FIG. 6A depicts a process for reconstructing surfaces in a physical space using a pose graph. Large-scale surface reconstruction involves registering range (depth) data from multiple source locations. The farther apart those source locations, the larger the error in co-registering them. Co-registration techniques such as iterative closest point solve the problem on small scales but result in undesirable artifacts, such as drift or twist, over large scales. One approach involves anchoring volumes to nodes in a pose graph and permitting overlap between adjacent volumes to avoid gaps. That solution benefits from knowledge of accumulated global tracking error, which is not always available or accurate. Furthermore, it uses mesh zippering to combine surfaces from all the disparate volumes.

A tracking-and-mapping solution, such as Simultaneous Localization and Mapping or Parallel Tracking and Mapping yield a pose graph whose nodes correspond to estimates of coordinates of "features" in the world and whose links correspond to transforms between those nodes. These are techniques to construct a map of an unknown environment

(or to update a map within a known environment) while simultaneously keeping track of a rig's location in the physical environment.

Both nodes and links contain errors that change over time as the pose graph is refined, e.g., due to further processing or incorporation of additional measurements. Each depth map (range data image) is assigned a pose (position and orientation) relative to a node in the pose graph. There is also a collection of fusion volumes, each of which is anchored to a node or nodes in the pose graph. As the pose graph evolves over time, the volumes migrate and, in general migrate, disparately. To address this issue, instead of using a single anchor for each volume, each vertex of each fusion volume can be anchored to a different pose graph node, in one approach. For example, a cubical fusion volume has eight vertices and therefore at least eight anchors. Furthermore, each fusion volume is placed perfectly aligned and adjacent to its neighbors so that adjacent volumes have a shared face with shared vertices. As the vertices migrate, they migrate in concert, meaning that all volumes remain perfectly flush—with no gaps or overlaps. This is equivalent to applying a free-form deformation to each volume. Moreover, the deformation can be applied either before or after fusion, or both before and after fusion, yielding a large-scale reconstruction that benefits from pose graph refinement.

Step 600 indicates that a rig moves in different poses in a physical space. For example, the rig can be a robot or an HMD device worn by a user that moves around a physical space, continuously capturing images of the physical space. Step 601 involves obtaining images of the physical space from a visible light camera on the rig. Step 602 involves obtaining orientation readings from an orientation sensor on the rig. Step 603 involves processing the images and orientation readings to provide and update a pose graph. For example, the pose graph can be updated from a first state to a second state. Step 604 obtaining orientation depth maps of volumes of the physical space from a depth sensor on the rig. A depth map indicates the distance from the sensor to an object in the physical space, and an orientation between the sensor and the object. A depth map can be obtained at different times, such as when the rig is aimed at an area of a physical space for which depth data has not been obtained.

Step 605 involves partitioning the physical space into volumes, and creating a new volume if appropriate. A volume can be a uniformly shaped space such as a cube, cuboid or other hexahedron in the physical space. Other polyhedron shapes could be used as well. In one approach, the volume is divided into sub-volumes referred to as blocks, where there are N×N×N blocks per volume, and N is an integer of one or more. Further, each block can be divided into 8×8×8 voxels.

Step 606 involves anchoring vertices of the volumes to points in the pose graph. Each point in the pose graph can be a node or a combination of nodes, for example. Generally, the vertices of a volume are anchored to at least two different points in the pose graph.

The anchoring of a vertex to a point in a pose graph means a position of the vertex in a physical space is defined based on a coordinate system of that point. The coordinate system can be that of a node, or one obtained as a linear combination of coordinate systems of multiple nodes. See FIG. 6C for further details.

Step 607 involves reconstructing surfaces in the physical space based on a state of the pose graph. Note that the some of the steps can be performed concurrently or in different orders. Generally, the obtaining of the images and the updating of the pose graph can occur separately from the

obtaining of the depth maps and the reconstructing of the surfaces. Similarly, the obtaining of the depth maps can occur separately from the obtaining of the images and the updating of the pose graph. As the rig continues to move, additional images and depth maps may be captured and processed. Further details regarding the process are provided below.

FIG. 6B depicts further details of step 603 of FIG. 6A. Step 603 involved processing images and orientation readings to provide and update a pose graph. Step 610 involves inputting an orientation reading and an image of a pose of the rig, and determining image features. Generally, an image can be processed to represent it in a compact way, to reduce data storage requirements and to facilitate feature-based image matching. An image can include features such as colors, patterns, shapes, lines and so forth, and common features in two images can be identified. One way to represent images is to use a feature detector and descriptor. Popular descriptors include the Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF). These are known to be robust to scale, rotation and illumination changes.

For instance, in the SIFT technique, keypoints of objects in a physical space are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of three or more features that agree on an object and its pose is then subject to further detailed model verification and, subsequently, outliers are discarded. Finally, the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

Decision step 611 determines whether the pose is new. In one approach, if the pose is not sufficiently different than a previous pose or another pose, it may not be considered to be new, in which case the flow returns to step 610. For example, the pose may be new if the orientation is significantly different than in other nodes. Or, the pose may be new if features of the image are significantly different than in other nodes or if a specified amount of time has passed since last new pose. In one approach, the camera provides a sequence of video images and selected images that are suitable for use in defining a node.

If the pose is new, step 612 adds a new node to the pose graph. Step 613 determines a coordinate system associated with the orientation of the pose. Or, the coordinate system may already have been determined Step 614 associates the image taken with the pose (e.g., the image captured by the camera on the rig when the rig/camera has assumed the pose) and the coordinate system with the new node. For example, this can involve storing the information in a database, as discussed further below. An image can be represented and stored in any form, including pixel data and/or extracted feature data. Step 615 determines a correspondence between the image associated with the new node and images of one or more other nodes in the pose graph. For instance, this can involve comparing the features of the image of the new node to features of images of one or more

other nodes which are close to the new node in the pose graph. The correspondence between two images is relatively high when the features in the two images are relatively similar. For example, the degree of correspondence can be determined using the SIFT technique.

Step **616** determines a transform between the coordinate system of the new node and the coordinate systems of the one or more other nodes. Step **617** associates the transform and the correspondence with one or more links between the new node and the one or more other nodes. Step **618** updates the other nodes and links in the pose graph if appropriate. For example, updating of the state of the pose graph can involve performing a bundle adjustment in which the coordinate systems and the transforms are adjusted, or updating one or more of the correspondences, or performing a relocalization in which there is a change in topology in the pose graph. For instance, the relocalization can comprise a loop closure, in which a closed loop of the nodes is formed in the pose graph.

FIG. **6C** depicts further details of step **606** of FIG. **6A**. Step **606** involved anchoring vertices of the volumes to points in the pose graph. Step **620** involves selecting a vertex of a volume. For a volume, a decision as to what point a vertex will be anchored to can be made separately for each vertex. Step **621** determines a closest node. One approach is to anchor a vertex to a closest node. For example, a distance of the vertex from each node can be determined, where the node with the smallest distance is selected. A distance between a vertex and a node can be defined as a distance of the vertex from the origin of the coordinate system of the node. See, e.g., FIG. **16A** for further details.

In one option, step **622** anchors the vertex to the closest node. At decision step **623**, if there is no next vertex in the volume to anchor, the process ends at step **624**. If there is a next vertex to anchor, step **620** is repeated by selecting the next vertex. In a second option, step **625** identifies one or more other nodes within a radius of the closest node. Step **626** determines a point which is a combination of the closest node and the one or more other nodes. For example, the point can be a central point between origins of coordinates systems of the closest node and the one or more other nodes.

For example, see FIG. **16B**. Step **627** anchors the vertex to the point. This approach can anchor a vertex to a point which is based on a combination of nodes, such as a combination of all nodes within a specified distance from a closest node. See, e.g., FIG. **16B** for further details. Once the closest node is determined, the nodes within a specified radius of the closest node can be identified. For example, this can be nodes for which the origin of their coordinate system is within a radius of the origin of the coordinate system of the closest node. The radius defines a sphere in which other nodes are located. In one approach, the radius is fixed. In another approach, the radius is a specified percentage of the distance between the closest node and the vertex. The radius or other distance could also be a function of the size of the volume such that the radius is relatively larger when the volume is relatively larger.

Anchoring a vertex to a point which is based on a combination of nodes can provide a smoothing effect when surface reconstruction is performed.

It is possible to use shapes other than a sphere to identify nodes which are relatively close to the closest node.

Once a node or combination of nodes is identified, the vertex is anchored to the identified node or combination of nodes, e.g., by updating a record in a database, as discussed further below. Thus, a vertex of a volume is associated with a node such that the vertex is positioned in the physical

space based on the coordinate system of the node. When the coordinate system of the node is adjusted, the position of the vertex of the volume in the physical space is also adjusted. Typically, a vertex of a volume continues to be anchored to one node, but it is possible for a vertex of volume to be re-anchored to a different node, such as if the pose of the different node becomes closer to the pose used to obtain the volume, due to updating of the pose graph. Also, the vertices can be anchored to the nodes on a one vertex-to-one node basis or a many vertices-to-one node basis. Similarly, the vertices can be anchored to points in the pose graph on a one vertex-to-one point basis or a many vertices-to-one point basis.

In some cases, there may be fewer nodes than vertices, such as when the pose graph is started. Initially, there could be only one node, in which case all vertices would be anchored to that node. As new nodes are added, the vertices and be re-anchored to the new nodes. The updating of the pose graph can comprise adding one or more nodes to the pose graph, wherein, for one or more volumes, a number of points in the pose graph to which the vertices of the volume are anchored increases during the adding of the one or more nodes to the pose graph.

FIG. **6D** depicts further details of step **607** of FIG. **6A**. Step **607** involved reconstructing surfaces in the physical space based on a state of the pose graph. Step **630** involves identifying one or more volumes and associated depth maps for surface reconstruction. For example, these volumes may correspond to a portion of a physical space in the user's field of view. As the user turns his head and looks at different portions of the physical space, the volumes involved in surface reconstruction can change accordingly. Step **631** involves determining one or more nodes to which the one or more volumes are anchored. This can be done by looking up a node which is cross referenced to a volume in a database. Step **632** involves determining the coordinate systems of the one or more nodes. This can be done by looking up a coordinate system which is cross referenced to a node in a database. Step **633** involves projecting the one or more volumes into the one or more depth maps based on the one or more coordinate systems. Essentially, the depth map is defined with respect to a coordinate system such that the depth values of the depth map are associated with portions of a volume based on the coordinate system.

FIG. **7** depicts a surface reconstruction pipeline for carrying out the process of FIG. **6A**. In this example implementation, the SR pipeline runs across both an Application-Specific Integrated Circuit (ASIC) **700** and a System-on-a-Chip (SoC) **740**. The ASIC and SoC each host multiple devices and subsystems. Arrows indicate a data flow direction. Sensors **710** connect to the ASIC. The sensor are carried on a rig such as an HMD device or a robot. These sensors include an orientation sensor **711** such as an inertial measurement unit (IMU) which senses rotation and linear acceleration. A visible light camera **712** obtains visible light images of a physical space and a depth sensor **713** obtains a depth map of a physical space. An example implementation uses a time of flight camera as the depth sensor. This sensor measures the round-trip time of light flashed from the camera, bounced off surfaces in the physical space, and returned to the sensor. The total round-trip time takes tens of nanoseconds. Another implementation of the depth sensor is a stereo pair of visible light cameras, but this approach may result in a lower quality depth map.

A tracking subsystem **720** converts sensor data into poses and a map of interconnected coordinate frames. For example, this can be a head tracking subsystem in the

example case of an HMD device. The tracking subsystem **720** is responsive to a map service subsystem **741**, and includes a tracking and mapping process **721**. The tracking subsystem **720** and the map service subsystem **741** together form a tracking and mapping subsystem **742**.

An SR Fusion and Extraction Kernel (SRFEK) **730** converts depth maps and poses into surface data. The SRFEK runs on the ASIC, and communicates with other processes in both the ASIC and the SoC. The SRFEK takes in two pieces of input data: depth maps and poses. A raw to depth subsystem **731** captures raw infrared images from the depth sensor and converts them into depth values in depth maps. The depth values indicates a distance of one or more surfaces in the physical space from the depth sensor. These depth maps have lens distortion. To address this, an undistortion subsystem **732** removes the effects of distortion from the lens of the camera from the depth images, to provide an undistorted depth map. This can involve applying a model of lens distortion, whose parameters are measured through calibration, to yield a flat undistorted image. At runtime, this can involve creating an undistortion map (a lookup table) and applying it to each depth map pixel.

A 3D fusion subsystem **734** combines the undistorted depth map with a pose to fuse it into a 3D volume. Fusion involves projecting depth maps (which are 2D images) into a volume (which is 3D). For example, a signed distance function (SDF) can be used which represents a distance from the depth sensor to the surface of an object. See FIGS. **8A** and **8B** for further details.

An extraction algorithm **733** creates a polygon mesh from that volumetric data. Not all applications require a polygon mesh, but they are often useful in applications such as game engines which use meshes for rendering and collision. One way to extract a mesh after populating the SDF volume uses an algorithm called Marching Cubes. The idea behind Marching Cubes involves turning each box of volumetric data into a set of polygons, where the vertices of those polygons lie on edges of the boxes.

Outputs of the 3D fusion subsystem **734** and the extraction algorithm **733** can be stored in a SRFEK cache **735**. This output includes the reconstructed surfaces which represent the geometry of the physical space.

A SR service **750** is a higher level process which controls SRFEK, giving it chunks of data and commands to process that data. The SR service also feeds surface data which is obtained from surface reconstruction to applications. The service includes a depth map acquisition subsystem **751** which decides when to trigger the acquisition of depth maps, based on where and when they are needed. A volume placement subsystem **752** decides how to partition the physical space into volumes, e.g., how many volumes to create and where to place them. Deciding where to place volumes is the initial problem. We can start with a set of previously-defined SR volumes. See FIGS. **9A** and **9B** for further details.

SR data can occupy a lot of memory, so the service decides which data to keep in memory, e.g., an SR service cache **754**, and which to store for later use, e.g., in a backing store **755**, using a cache eviction and replenishment subsystem **753**. The SR service decides what SR data to evict and replenish, when to evict and replenish the data, and supplies applications with SR data. The SR service is useful in managing scale, both in terms of the size of the physical space and the quantity of data which is generated. The system should work on physical scales that extend beyond the immediate vicinity of the user, e.g., in physical spaces of several meters such as in an office building. Brute-force

solutions will not work because even modestly-sized regions uses hundreds or thousands of SR volumes and each SR volume can use a few megabytes. The SR system can scale by partitioning a space into volumes and paging (e.g., transferring) those volumes between memory and disk.

The SR Service can accomplish both persistence and data scalability with the cache eviction and replenishment subsystem. For example, this subsystem can establish a maximum number of volumes. This provides a cap on the amount of SoC main memory the SR service consumes. When that cap would be exceeded by adding more data, or more SR volumes, we can evict some other volumes. This is analogous to a least-recently-used (LRU) eviction policy, except the SR Service can evict the farthest volumes, e.g., farthest from the rig. Moreover, when the rig re-enters a region for which it previously captured SR data, the cache can be replenished with that data. Thus, we can intermittently replenish the cache from the backing store by loading the closest volumes to the rig. Note that new volumes can still be added so they also contribute to the cache population.

Many applications can consume SR data in similar ways. An SR extensions and utilities subsystem **760** provides common functionality for application developers. Extensions and utilities can include occlusion rendering **761** for occluding holograms with real-world geometry, mesh post-processing **762** to add useful auxiliary information, collision detection **763** between real and virtual objects, play space definition **764** and negative space allocation **765** to help with hologram placement. Examples of SR mesh post-processing routines include welding vertices, zippering meshes, computing surface normals, clipping triangles from overlapping meshes, smoothing meshes, and simplifying meshes to reduce the number of polygons.

Regarding hologram placement, meshes can be too fine and complicated for certain kinds of hologram placement logic. Instead, application developers can find it easier to deal with abstractions such as walls, ceilings and floors. One solution is to find flat triangles, identify connected flat regions, and then then fit planes to those regions. To find contiguous regions of low curvature, multiple separate meshes are first combined into a single contiguous mesh. This is where welding and zippering are useful.

Regarding computing surface normals, to find flat regions, it is useful to compute surface curvature. The SR pipeline can generate per-vertex normals, either from the SDF (using finite differences) or from the mesh (using per-face normals).

Another example of SR mesh post-processing is applying texture as captured from the camera.

FIG. **8A** depicts an example of a signed distance function (SDF) as discussed in connection with the 3D fusion subsystem **734** of FIG. **7**. The horizontal axis depicts a distance from the depth sensor to an object **800** in a physical space. The vertical axis depicts a value of the SDF. The object has a front surface **801** and a back surface **802**. A line **805** represents the SDF. Regions outside the object have a positive distance, regions inside the object to have a negative distance, and regions at a surface of the object have a value of zero. In some cases, the front surface of an object but not the back surface is detected. Fusion involves populating a volume of voxels with SDF values inferred from projecting voxels into depth maps. We start with an SDF volume, then acquire a depth map and its pose relative to that volume. The fusion algorithm injects a posed depth map as follows. For each voxel, project it into the depth map and update the SDF value according to the distance of the voxel relative to the distance from the surface. Surfaces lie on the boundary

between negative and positive SDF values, where the function is zero—the so-called zero level set. This procedure is repeated for each depth map.

FIG. 8B depicts an example of a depth map projected into voxels. Each square represents a voxel. The voxels shown can be part of one or more blocks. A solid line **810** represents a surface of an object. A dashed line **811** represents a surface which is not detected but is assumed to be far from the sensor, past the maximum depth which can be sensed. Depth values for this surface can be set at a default depth. Voxels traversed by these lines have SDF=0 and are represented by a diagonal lined pattern, such as with voxel **813**. Voxels which are considered to be within the object have SDF<0 are represented by a dotted pattern, such as with voxel **812**. Voxels which are considered to be outside the object have SDF>0 and are represented by unpatterned boxes, such as with voxel **814**. A line **815** represents a boundary of a view frustum of the depth sensor. The view shown can be an overhead view of a room, where the line **810** represents a wall and a frame on the wall.

FIG. 9A depicts a set of volumes **900-905** and a frustum **920** of a depth sensor. Note that some overlap between the volumes is shown. Some of the volumes are shown with dashed line patterns to clearly distinguish the volumes from one another. When the SR service decides to acquire a depth map, the depth sensor is activated. The frustum represents the field of view of the depth sensor, e.g., a portion of the physical space in which depth readings can be obtained. The SR service then compares the depth map frustum with the set of SR volumes. If any portion of the depth map frustum is not covered by any SR volume, the service can create a new volume. However, the placement of the new volume, including its orientation relative to the other volumes, is an issue. The new volume can be initially placed somewhere where the frustum lacks coverage by any existing volume. Then, the new volume is repositioned to align with the existing volumes and share vertices with them. The repositioning can involve a translation and/or a rotation in the physical space which refines the new volume's position.

FIG. 9B depicts a new volume which is added to the set of volumes **900-905** of FIG. 9A, as represented by an initially-placed volume **907** and a repositioned volume **906**.

FIG. 10 depicts an example of a pose graph **1000** which represents poses of a rig in a physical space. A pose graph includes nodes connected by links. A node comprises an image of a physical space as captured by a visible light camera, and an associated coordinate system. A link (e.g., edge) between two nodes comprises a correspondence between the images of the two nodes and a transform between the coordinate frames of the two nodes. The pose graph represents a path cycle of the rig. Arrow **1210** depicts a path of the rig. Examples nodes are N1-N9. Example links are a link L(N1,N2) which is between nodes N1 and N2 and a link L(N1,N3) which is between nodes N1 and N3.

In particular, a pose graph of a rig can be used to optimize the placement of vertices of volumes in a physical space for surface reconstruction. The problem of spatial partitioning is not specific to SR, and there are many spatial partitioning schemes. To understand why the techniques described herein work well for SR, first consider various aspects of how rig tracking works. The tracking and mapping subsystem **742** (FIG. 7) of a rig tracks the rig's trajectory through space and time by analyzing images (e.g., pictures) which are captured by the camera of the rig. Imagine the rig powers on for the first time and the camera captures images. The tracking and mapping subsystem extracts features from the images, estimates the locations of the features in the images and stores

the features. The tracking and mapping subsystem also creates a coordinate system (e.g., a coordinate frame) it will use to measure positions and orientations. Both the images and the coordinate system are stored together with a node, in a map. Then the rig moves. The tracking and mapping subsystem takes more pictures and creates another coordinate frame that the map associates with another node. The map service subsystem **741** recognizes that first and second images have some common features. The mapper uses computer vision algorithms and IMU data to estimate the relative poses between those two nodes.

The link between nodes represents correspondences between images and a transform between the coordinate systems of the two nodes. However, when the mapper can obtain multiple estimates for the transform between nodes, those transforms can be inconsistent. The tracking and mapping system minimizes the inconsistency by adjusting the coordinate transforms and feature locations. This is called bundle adjustment. As the rig moves around, the mapper continues to add nodes and links, and this results in additional pose adjustments. As a result, the map of the physical space changes over time. That is, estimates of feature positions move and the pose graph is refined. For example, the links in the pose graph can change distance or orientation. SR should be robust against pose adjustments.

To understand how severe this problem can be and how far a feature in an image can appear to move, consider loop closure—an abrupt form of pose graph adjustment. Loop closure is a form of relocalization. For example, imagine the rig visits separate spaces without visiting the regions that connect them. The mapper will have separate map components for each space. Then, imagine the rig visits the regions between the spaces. The tracking and mapping subsystem will recognize that the regions are connected, and will add links to the pose graph. Relocalization refers to any topological change to the pose graph.

One form of loop closure occurs when the rigs ends up at a point in space where it already visited, and the tracking and mapping subsystem observes that fact. For example, imagine a user wearing an HMD device walks into a room such as in an art gallery and always faces the wall nearest to the user, beginning in a starting position. The user does not look across the room or behind the user. The user then circumnavigates the room, looking at each wall in turn, as the tracking and mapping subsystem creates new nodes. Just before the user returns to the starting position, the tracking and mapping subsystem does not yet recognize that it is at the starting point again. But, eventually, the tracking and mapping subsystem will recognize features captured while the user was at the starting point and will identify correspondences between images taken at the beginning and end of this loop. It will therefore create a new link in the pose graph, thus forming a closed loop. Just before loop closure, the transform between the start and end of the loop entails many hops. Just after loop closure, the transform between the start and end of the loop entails only a single hop. Each hop has some error, and the errors accumulate. The amount of accumulated errors is smaller just after loop closure. So, just before and just after loop closure, the transforms between nodes will abruptly change. The techniques provided herein allow the SR process to benefit from adjustments to the pose graph by anchoring volumes to the nodes and their coordinate systems in a pose graph. When the pose graph changes, the SR volumes come along for the ride since the vertices of the volumes follow the pose graph node to which they are anchored.

FIG. 11A depicts an example data record for an image captured by a camera. Each image, or a set of images such as with video, can be stored in a data record, e.g., Image1. An orientation reading, e.g., OR1, of the rig can be associated with the image.

FIG. 11B depicts an example data record for a depth map captured by a depth sensor. Each depth map can be stored in a data record, e.g., DM1. An orientation reading, e.g., OR1, of the rig can be associated with the depth map.

FIG. 12A depicts an example data record of a pose graph. A record for a node may include a node identifier, e.g., N1, a camera image, e.g., Image1 and a coordinate system, e.g., CS1. A record for the link L(N1,N2) may include a link identifier, e.g., L(N1,N2), image correspondence data, e.g., C(N1,N2), and a transform between coordinate systems, e.g., T(N1,N2). A record for the link L(N1,N3) may include a link identifier, e.g., L(N1,N3), image correspondence data, e.g., C(N1,N3), and a transform between coordinate systems, e.g., T(N1,N3). A transform can be a matrix which relates one coordinate system as a matrix to another coordinate system as a matrix.

FIG. 12B depicts an example data record of a first volume in which each vertex of the volume is anchored to a different node. For example, this can be Vol1, discussed also in connection with FIGS. 15 and 16A-16E. The record include a volume identifier, e.g., Vol1, and an entry for each vertex of the volume. Each entry includes a vertex identifier and an identifier of a point to which the vertex is anchored. For example, vertices v1, v2, v3, v4, v5, v6, v7 and v8 are anchored to nodes N1, N2, N3, N4, N5, N6, N7 and N8, respectively.

FIG. 12C depicts an example data record of a second volume which is adjacent to the first volume, and in which each vertex of the volume is anchored to a different node. For example, this second volume can be Vol2, discussed also in connection with FIGS. 15 and 16A-16E. For example, vertices v2, v4, v6, v8, v9, v10, v11 and v12 are anchored to nodes N2, N4, N6, N8, N9, N10, N11 and N12, respectively. As seen, Vol1 and Vol2 have common nodes v2, v4, v6 and v8. These common nodes define a common face of these two adjacent volumes.

FIG. 12D depicts an example data record of a first volume in which one vertex of the volume is anchored to a combination of nodes. Here, v6 is anchored to a combination of N5, N6 and N11.

FIG. 12E depicts an example data record of a second volume in which one vertex of the volume is anchored to a combination of nodes. Consistent with FIG. 12D, v6 is anchored to a combination of nodes N10, N11 and N12.

FIG. 13A depicts an example physical space, showing a frustum of a depth sensor. A physical space 1300 is a room such as in the home of a user 1301 wearing an HMD device 1302 as an example of a rig. The depth sensor on the HMD device has a field of view which is represented by a frustum 1303 extending between a back plane 1304 and a front plane 1305 at a wall 1306 of the room. The wall includes a doorway 1307 and a decorative painting 1308. The painting is attached to the wall 1306 and has a noticeable depth. A line 1309 represents a central axis of the field of view of the depth sensor, e.g., the direction in which the depth sensor (and the user and the HMD device) is looking. A Cartesian coordinate system 1310 with orthogonal axes of xr, yr and zr is an example coordinate system of the HMD device (r denotes the rig). For example, the xr axis can have the same direction as a vector (coincident with the line 1309) which represents the orientation of the rig. The yr axis can be defined to extend parallel to the ground. The zr axis is then

orthogonal to the xr and yr axes. A Cartesian coordinate system 1311 of the physical space with orthogonal axes of x, y and z is also depicted. The depth sensor is at an origin of the coordinate system.

FIG. 13B depicts a depth map 1315 consistent with FIG. 13A. The depth map includes a region 1320 for the wall 1306, a region 1321 for the painting 1308 and a region 1322 for the doorway 1307.

FIG. 13C depicts distance values of the depth map of FIG. 13B. The horizontal axis depicts a direction which is orthogonal to the center line 1309 of the depth sensor, such as a direction of the yr axis. The vertical axis depicts a distance of the object from the depth sensor, such as a distance along the xr axis. A line 1330 represents the depth. Portions of the line which correspond to a depth of the wall have a depth value of Dwall. A portion of the line which corresponds to a depth of the painting has a depth value of Dpainting. A portion of the line which corresponds to a depth of the doorway has a depth value of Dmax, a maximum depth which can be detected by the depth sensor. In this case, since the depth sensor looks through the doorway to a surface which is beyond the detection range, the depth value can be set to a maximum value. Dmin represents a minimum depth which can be detected by the depth sensor.

FIG. 14A depicts the example physical space of FIG. 13A, showing a field of view of a visible light camera. Generally, a field of view 1340 of the camera can differ from the field of view of the depth sensor, although they can look in the same direction, e.g., the direction of the HMD device, and they can be substantially overlapping. In this example, the field of view 1340 of the camera is slightly narrower and taller than the field of view of the depth sensor. The coordinate system 1310 of the HMD device is the same as in FIG. 13A as an example but this is not required. The axis can be coincident with a center line of the camera which is the same as the center line 1309 of the depth sensor.

FIG. 14B depicts an image obtained by the visible light camera in the pose of FIG. 14A. The image 1400 includes a portion 1401 which represents the wall, a portion 1402 which represents the painting, and a portion 1403 which represents the doorway.

FIG. 14C depicts the example physical space of FIG. 13A, showing another field of view of the visible light camera as a pose of the HMD device is changed. In this case, the user has moved such that the camera on the HMD device captures a different image of the physical space. For example, the user may have rotated his head and/or walked around in the room. The field of view 1420 is defined relative to the coordinate system 1430 which differs from the coordinate system 1310 of FIG. 14A. In one approach, the xr axis is coincident with a center line 1421 of the camera.

FIG. 14D depicts an image obtained by the visible light camera in the pose of FIG. 14C. The image 1440 includes a portion 1441 which represents the wall 1306 and the adjacent wall 1415, and a portion 1442 which represents the painting. As can be seen, the images 1400 and 1440 will have corresponding features such as the horizontal lines 1444 of the frame of painting, the vertical lines 1443 of the frame of the painting, and the colors and shapes of the flower and vase in the painting.

FIG. 15 depicts an initial placement of volumes in the physical space of FIG. 13A. The view frustum 1303 is depicted along with eight volumes (including example volume Vol1 of FIG. 12B and Vol2 of FIG. 12C) which are positioned to encompass the view frustum. The volumes are initially positioned adjacent to one another so there are no gaps or overlaps between them. The vertices of the volumes

are defined with respect to one or more nodes or combinations of nodes, as discussed. The volumes in this example are cubes and are initially placed with a bottom surface parallel to the ground.

FIG. **16**A depicts an example of step **606** of FIG. **6**A involving the volumes Vol**1** and Vol**2** of FIG. **15**, in which each vertex of a volume is linked to a different node, consistent with the data records of FIGS. **12**B and **12**C. Vol**1** includes vertices labelled as v**1**-v**8**. Vol**2** includes vertices labelled as v**2**, v**4**, v**6** and v**8** (shared with Vol**1**) and v**9**, v**10**, v**11** and v**12**. In this example, there are only two volumes for simplicity, but others could be used as well. The anchoring of some of the vertices is shown as an example. For example, v**5** is anchored to node N**5** and its coordinate system **1610** which has the origin O**5** and axes x**5**, y**5** and z**5**. Each axis can be considered to be a vector which has a direction in a physical space. V**6** is anchored to node N**6** and its coordinate system **1620** which has the origin O**6** and axes x**6**, y**6** and z**6**. V**11** is anchored to node N**11** and its coordinate system **1630** which has the origin O**11** and axes x**11**, y**11** and z**11**. V**9** is anchored to node N**9** and its coordinate system **1640** which has the origin O**9** and axes x**9**, y**9** and z**9**. Generally, the z axes of the different coordinate system can be parallel to one another and represent height. The x and y axes can differ among the coordinate systems.

FIG. **16**B depicts an example of step **606** of FIG. **6**A involving the volumes Vol**1** and Vol**2** of FIG. **15**, in which a vertex of a volume is linked to combination of nodes, consistent with the data records of FIGS. **12**D and **12**E. v**5**, v**11** and v**9** are anchored to N**5**, N**11** and N**9** as in FIG. **16**A. However, v**6** is anchored to a point **1650**, represented by a square. The point is the origin Oc of a coordinate system **1660** which includes the axes xc, yc and zc. The coordinate system is shown separately for clarity. In this example, N**6** is first identified as the node among all existing nodes in the pose graph which is closest to v**6**. Subsequently, N**5** and N**11** are identified as nodes within a radius R (a specified distance) of N**6**. For example, the origin Oc can be defined as a central point between O**5**, O**6** and O**11**, giving equal weight to each origin/node. This approach provides an equal weight for all of the points within the radius.

FIG. **16**C**1** depicts distance d**5** and d**11** of the nodes N**5** and N**11**, respectively, from the closest node N**6** to v**6**. Another approach is to obtain the central point by assigning different weights to the different nodes within the radius. For example, the point of the closest node (e.g., O**6** of N**6**) can be given a higher weight than the other points within the radius. The other points within the radius could be weighted equally or be given a decreasing weight in proportion to an increasing distance from the point of the closest node. In another approach, the closest nodes and the other nodes within the radius are weighted in inverse proportion to their distance from the vertex.

The point **1650** represents a center point between the nodes N**5**, N**6** and N**11** (with origins O**5**, O**6** and O**11**, respectively) giving each node or origin equal weight. In another approach, the point **1651** represents a center point between the nodes N**5**, N**6** and N**11** (with origins O**5**, O**6** and O**11**, respectively) giving the closest node (N**6**) to v**6** a higher weight than N**5** and N**11**.

FIG. **16**C**2** depicts an x-axis of the combined coordinate system **1660** of FIG. **16**B. The coordinate system **1660** is obtained as a linear combination of the coordinate systems of N**5**, N**6** and N**11**. For example, in the case of all nodes being weighted equally, the x-axis, xc can be obtained by vector addition of the x-axes, x**5**, x**6** and x**11**. Similarly, the

y-axis, yc can be obtained by vector addition of the y-axes, y**5**, y**6** and y**11**, and the z-axis, zc can be obtained by vector addition of the z-axes z**5**, z**6** and z**11**. Or, if all z-axes are defined to be vertical, no vector addition is needed to obtain zc. In the case of the nodes being weighted unequally, the axes of the coordinate system **1620** can be obtained by weighting the axes in the vector addition accordingly.

FIG. **16**D depicts depth maps of undistorted objects within the volumes Vol**1** and Vol**2** when these volumes are not distorted. The depth maps include a region **1320** for a wall, a region **1321** for a painting and a region **1322** for a doorway, as in FIG. **13**B. The regions are provided within the volumes Vol**1** and Vol**2**. Vol**1** has a height of h**2** and a width w**2**. Vol**2** also has this height and width. The region **1321** of the painting has a height h**1** and a width w**1**.

FIG. **16**E depicts depth maps of distorted objects within the volumes Vol**1** and Vol**2** when these volumes are distorted. In this example, an adjustment is made to the node to which v**2** is anchored, resulting in a change in position of v**2** from a location **1661** to a location **1662**. This represents a change in the height of dh**2** and a change in the width of dw**2**. Thus, Vol**1** and Vol**2** undergo a change in shape, or a distortion. The regions of the depth maps within the volumes are also distorted in correspondence with the distortion of the volumes. For example, the undistorted wall region **1320** becomes the distorted wall region **1320**d, the undistorted painting region **1321** becomes the distorted painting region **1321**d, and the undistorted doorway region **1322** becomes the distorted doorway region **1322**d. For example, the right upper corner of the painting can change from the point **1665** to the point **1666**. The right upper corner can move higher by a distance dh**1** which is proportional to h**1**×dh**2**/h**2**, and to the right by a distance dw**1** which is proportional to w**1**×dw**2**/w**2**. These are approximations which depict the general concept of distorting the 3D object within a volume based on distortion of the volume. In one approach, a bilinear interpolation can be performed for any point inside the volume, to determine the distorted shape of the object.

FIG. **17** depicts an example application in which a virtual object is placed in the physical space of FIG. **13**A. As mentioned, surface reconstruction can be used in many applications, including placing a virtual object, defining a play space, path-finding, collision detection or occlusion of virtual objects a physical space. In this example, a virtual object **1710** is a person which is visible to the user **1301** as an augmented reality image of the HMD device **1302**. A field of view **1700** is associated with the augmented reality projection system of the HMD device. The virtual object can be placed in an appropriate location in the physical space **1300** since the surfaces in the room are known. For example, the virtual object can be placed a specified distance in front of the wall **1306** and to the side of the doorway **1307**. Moreover, real-world objects which are behind the virtual object such as the painting **1308** can be occluded to provide greater realism. Note that different or common fields of view can be associated with the depth sensor, the camera and the augmented reality projection system. Further, while example implementations involving an HMD device have been described, the techniques provided herein are applicable to many other scenarios, including those involving robots and other types of rigs.

Accordingly, it can be seen that, in one aspect, a method for reconstructing a physical space comprises: obtaining a plurality of images of the physical space using a camera carried by a rig while the rig moves in the physical space through a plurality of poses; determining an orientation and a coordinate system of each pose; partitioning the physical

space into a plurality of volumes, each volume of the plurality of volumes comprises vertices; for each volume of the plurality of volumes, obtaining a depth map of the physical space using a depth sensor carried by the rig; providing a pose graph in a first state, the pose graph in the first state comprising a plurality of nodes and links between nodes of the plurality of nodes, each node of the plurality of nodes is associated with: one of the poses, one of the images associated with the one of the poses and one of the coordinate systems associated with the one of the poses, and the links define correspondences between the images and transforms between the coordinate systems; for each volume of the plurality of volumes, anchoring the vertices of the volume to at least two different points in the pose graph in the first state, the vertices of the volume comprise shared vertices, the shared vertices are shared with an adjacent volume of the plurality of volumes when the pose graph is in the first state; updating the pose graph to a second state, the updating comprising updating one or more of the nodes and one or more of the links; and for each volume of the plurality of volumes, maintaining the shared vertices when the pose graph is in the second state, wherein the maintaining the shared vertices results in changing a shape of at least one volume of the plurality of volumes.

The method further comprises reconstructing a surface in the at least one volume of the plurality of volumes according to the second state of the pose graph; and during the reconstructing, for the at least one volume of the plurality of volumes, distorting a depth map of the physical space based on the shape of the at least one volume of the plurality of volumes.

Each node of the plurality of nodes can be associated with a location of the rig in the physical space; and for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph can comprise anchoring one vertex of the volume to one of the nodes in the pose graph for which the location is closest to the one vertex.

Each node of the plurality of nodes can be associated with a location of the rig in the physical space; and for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph can comprise anchoring one vertex of the volume to a combination of nodes in the pose graph, the combination of nodes can comprise one of the nodes in the pose graph for which the location is closest to the one vertex.

The combination of nodes can comprise one or more nodes within a specified distance of the one of the nodes in the pose graph for which the location is closest to the one vertex.

In the combination of nodes, each node of the one or more nodes within the specified distance can be weighted according to distance of the node to the one of the nodes in the pose graph for which the location is closest to the one vertex.

For one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph can comprise anchoring one vertex of the volume to one node in the pose graph and anchoring another vertex of the volume to another node in the pose graph.

The method further comprises, for one or more volumes of the plurality of volumes, anchoring each vertex of the volume to a different point in the pose graph.

The updating the pose graph can comprise adding one or more nodes to the pose graph; and for one or more volumes of the plurality of volumes, a number of points in the pose graph to which the vertices of the volume are anchored increases during the adding the one or more nodes to the pose graph.

The plurality of volumes can be non-overlapping with one another and gaps are not present between the volumes.

In another aspect, an apparatus for reconstructing a physical space comprises: an orientation sensor; a depth sensor; a camera; and a processor in communication with the orientation sensor, the depth sensor and the camera. The processor is adapted to: obtain a plurality of images of the physical space from the camera, associate a pose and a coordinate system with each image based on the orientation sensor, partition the physical space into a plurality of volumes, each volume of the plurality of volumes comprises vertices, for each volume of the plurality of volumes, obtain a depth map of the physical space from the depth sensor, provide a pose graph in a first state comprising a plurality of nodes and links between the nodes, each node of the plurality of nodes is associated with: one of the poses, one of the images associated with the one of the poses and one of the coordinate systems associated with the one of the poses, and the links define correspondences between the images and transforms between the coordinate systems, for each volume of the plurality of volumes, anchor the vertices of the volume to at least two different points in the pose graph in the first state, the vertices of the volume comprise shared vertices, the shared vertices are shared with an adjacent volume of the plurality of volumes when the pose graph is in the first state, update the pose graph to a second state, using an update of one or more of the nodes and one or more of the links, for each volume of the plurality of volumes, to maintain the shared vertices when the pose graph is in the second state, change a shape of at least one volume of the plurality of volumes, and reconstruct a surface in one or more of the volumes according to the second state of the pose graph.

In another aspect, a processor-readable memory comprises processor-readable software embodied thereon for programming a processor to perform a method for reconstructing a physical space, where the method comprises: partitioning a physical space into a plurality of volumes, each volume of the plurality of volumes comprises vertices; for each volume of the plurality of volumes, obtaining a depth map of the physical space; providing a pose graph in a first state, the pose graph in the first state comprising a plurality of nodes and links between nodes of the plurality of nodes; for each volume of the plurality of volumes, anchoring the vertices of the volume to at least two different points in the pose graph in the first state, the vertices of the volume comprise shared vertices, the shared vertices are shared with an adjacent volume of the plurality of volumes when the pose graph is in the first state; and updating the pose graph to a second state, the updating comprising updating one or more of the nodes and one or more of the links; and for each volume of the plurality of volumes, maintaining the shared vertices when the pose graph is in the second state, wherein the maintaining the shared vertices results in changing a shape of at least one volume of the plurality of volumes.

Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

What is claimed is:

1. A method for reconstructing a physical space, comprising:

obtaining a plurality of images of the physical space using a camera carried by a rig while the rig moves in the physical space through a plurality of poses;

determining an orientation and a coordinate system of each pose;

partitioning the physical space into a plurality of volumes, each volume of the plurality of volumes comprises vertices;

for each volume of the plurality of volumes, obtaining a depth map of the physical space using a depth sensor carried by the rig;

providing a pose graph in a first state, the pose graph in the first state comprising a plurality of nodes and links between nodes of the plurality of nodes, each node of the plurality of nodes is associated with: one of the poses, one of the plurality of images associated with the one of the poses and one of the coordinate systems associated with the one of the poses, and the links define correspondences between the plurality of images and transforms between the coordinate systems;

for each volume of the plurality of volumes, anchoring the vertices of the volume to at least two different points in the pose graph in the first state, the vertices of the volume comprise shared vertices, the shared vertices are shared with an adjacent volume of the plurality of volumes when the pose graph is in the first state;

updating the pose graph to a second state, the updating comprising updating one or more of the nodes and one or more of the links; and

for each volume of the plurality of volumes, maintaining the shared vertices when the pose graph is in the second state, wherein the maintaining the shared vertices results in changing a shape of at least one volume of the plurality of volumes.

2. The method of claim 1, further comprising:

reconstructing a surface in the at least one volume of the plurality of volumes according to the second state of the pose graph; and

during the reconstructing, for the at least one volume of the plurality of volumes, distorting a depth map of the physical space based on the shape of the at least one volume of the plurality of volumes.

3. The method of claim 1, wherein:

each node of the plurality of nodes is associated with a location of the rig in the physical space; and

for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph comprises anchoring one vertex of the volume to one of the nodes in the pose graph for which the location is closest to the one vertex.

4. The method of claim 1, wherein:

each node of the plurality of nodes is associated with a location of the rig in the physical space; and

for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph comprises anchoring one vertex of the volume to a combination of nodes in the pose graph, the combination of nodes comprises one of the nodes in the pose graph for which the location is closest to the one vertex.

5. The method of claim 4, wherein:

the combination of nodes comprises one or more nodes within a specified distance of the one of the nodes in the pose graph for which the location is closest to the one vertex.

6. The method of claim 5, wherein:

in the combination of nodes, each node of the one or more nodes within the specified distance are weighted according to distance of the node to the one of the nodes in the pose graph for which the location is closest to the one vertex.

7. The method of claim 1, wherein:

for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph comprises anchoring one vertex of the volume to one node in the pose graph and anchoring another vertex of the volume to another node in the pose graph.

8. The method of claim 1, further comprising:

for one or more volumes of the plurality of volumes, anchoring each vertex of the volume to a different point in the pose graph.

9. The method of claim 1, wherein:

the updating the pose graph comprises adding one or more nodes to the pose graph; and

for one or more volumes of the plurality of volumes, a number of points in the pose graph to which the vertices of the volume are anchored increases during the adding the one or more nodes to the pose graph.

10. The method of claim 1, wherein:

the plurality of volumes are non-overlapping with one another and gaps are not present between the volumes.

11. An apparatus for reconstructing a physical space, comprising:

an orientation sensor;

a depth sensor;

a camera; and

a processor in communication with the orientation sensor, the depth sensor and the camera, the processor is adapted to:

obtain a plurality of images of the physical space from the camera,

associate a pose and a coordinate system with each image based on the orientation sensor,

partition the physical space into a plurality of volumes, each volume of the plurality of volumes comprises vertices,

for each volume of the plurality of volumes, obtain a depth map of the physical space from the depth sensor,

provide a pose graph in a first state comprising a plurality of nodes and links between the nodes, each node of the plurality of nodes is associated with: one of the poses, one of the plurality of images associated with the one of the poses and one of the coordinate systems associated with the one of the poses, and the links define correspondences between the plurality of images and transforms between the coordinate systems,

for each volume of the plurality of volumes, anchor the vertices of the volume to at least two different points in the pose graph in the first state, the vertices of the volume comprise shared vertices, the shared vertices are shared with an adjacent volume of the plurality of volumes when the pose graph is in the first state,

update the pose graph to a second state, using an update of one or more of the nodes and one or more of the links,

for each volume of the plurality of volumes, to maintain the shared vertices when the pose graph is in the second state, change a shape of at least one volume of the plurality of volumes, and

reconstruct a surface in one or more of the volumes according to the second state of the pose graph.

12. The apparatus of claim 11, wherein the processor is adapted to:

during the reconstructing of the surface, for the at least one volume of the plurality of volumes, distort a depth map of the physical space based on the shape of the at least one volume of the plurality of volumes.

13. The apparatus of claim 11, wherein:

each node of the plurality of nodes is associated with a location in the physical space; and

for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph comprises anchoring one vertex of the volume to one of the nodes in the pose graph for which the location is closest to the one vertex.

14. The apparatus of claim 11, wherein:

each node of the plurality of nodes is associated with a location in the physical space; and

for one or more volumes of the plurality of volumes, the anchoring the vertices of the volume to at least two different points in the pose graph comprises anchoring one vertex of the volume to a combination of nodes in the pose graph, the combination of nodes comprises one of the nodes in the pose graph for which the location is closest to the one vertex.

15. The apparatus of claim 11, wherein:

the plurality of volumes are non-overlapping with one another and gaps are not present between the volumes.

16. An apparatus, comprising:

a storage device comprising code; and

a processor configured to execute the code to:

partition a physical space into a plurality of volumes, each volume of the plurality of volumes comprises vertices;

for each volume of the plurality of volumes, obtain a depth map of the physical space;

provide a pose graph in a first state, the pose graph in the first state comprising a plurality of nodes and links between nodes of the plurality of nodes;

for each volume of the plurality of volumes, anchor the vertices of the volume to at least two different points in

the pose graph in the first state, the vertices of the volume comprise shared vertices, the shared vertices are shared with an adjacent volume of the plurality of volumes when the pose graph is in the first state;

update the pose graph to a second state, the updating comprising updating one or more of the nodes and one or more of the links; and

for each volume of the plurality of volumes, maintain the shared vertices when the pose graph is in the second state, wherein the maintaining the shared vertices results in changing a shape of at least one volume of the plurality of volumes.

17. The apparatus of claim 16, wherein:

the plurality of volumes are non-overlapping with one another and gaps are not present between the volumes.

18. The apparatus of claim 16, wherein the processor is configured to execute the code to:

reconstruct a surface in the at least one volume of the plurality of volumes according to the second state of the pose graph; and

during the reconstructing, for the at least one volume of the plurality of volumes, distort a depth map of the physical space based on the shape of the at least one volume of the plurality of volumes.

19. The apparatus of claim 16, wherein:

each node of the plurality of nodes is associated with a location in the physical space; and

for one or more volumes of the plurality of volumes, to anchor the vertices of the volume to at least two different points in the pose graph, the processor is configured to execute the code to anchor one vertex of the volume to one of the nodes in the pose graph for which the location is closest to the one vertex.

20. The apparatus of claim 16, wherein:

each node of the plurality of nodes is associated with a location in the physical space; and

for one or more volumes of the plurality of volumes, to anchor the vertices of the volume to at least two different points in the pose graph, the processor is configured to execute the code to anchor one vertex of the volume to a combination of nodes in the pose graph, the combination of nodes comprises one of the nodes in the pose graph for which the location is closest to the one vertex.

* * * * *